Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1982

Aircraft state estimation for a ground directed bombing system.

Jauregiu, John A.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/20042

Downloaded from NPS Archive: Calhoun

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

LIBRARY, NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AIRCRAFT STATE ESTIMATION FOR A GROUND DIRECTED BOMBING SYSTEM

by

John A. Jauregui

December 1982

Thesis Advisor: H. A. Titus

Approved for public release, distribution unlimited

T208001

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Aircraft State Estimation for a Ground Directed Bombing System		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John A. Jauregui		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE December 1982
		13. NUMBER OF PAGES 96
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Kalman Filter Application
Ground Directed Bombing System
Adaptive Kalman Identifier
Radar Tracking

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The performance of a modified linear Kalman filter with adaptation is compared with that of a common adaptive alpha-beta filter for state estimation of a pilot controlled, ground directed bombing system. Of particular concern, is the accuracy and response of the alternative filters when the aircraft conducts random maneuvers in the vicinity of the target. The desirability of including deterministic forcing in the filter model is discussed and a technique utilizing an adaptive Kalman identifier to establish the pilot response to ground control heading commands is presented.

Approved for public release, distribution unlimited

Aircraft State Estimation for a Ground Directed Bombing System

by

John A. Jauregui
Captain, United States Marine Corps
B.S., United States Naval Academy, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1982

LIBRARY, NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940

ABSTRACT

The performance of a modified linear Kalman filter with adaptation is compared with that of a common adaptive alpha-beta filter for state estimation of a pilot controlled, ground directed bombing system. Of particular concern is the accuracy and response of the alternative filters when the aircraft conducts random maneuvers in the vicinity of the target. The desirablity of including deterministic forcing in the filter model is discussed and a technique utilizing an adaptive Kalman identifier to establish the pilot response to ground control heading commands is presented.

TABLE OF CONTENTS

I. INTRODUCTION 9

II. GROUND DIRECTED BOMBING SYSTEM (GDBS) SIMULATION . 11

A. COORDINATE SYSTEMS 11

B. AIRCRAFT DYNAMICS MODEL 13

C. GDBS MODEL IMPLEMENTATION 20

III. AIRCRAFT STATE ESTIMATION 22

A. BACKGROUND 22

B. GENERAL DESCRIPTION OF THE ALPHA-BETA FILTER . 22

C. GENERAL DESCRIPTION OF THE KALMAN FILTER . . . 23

D. FILTER ORDER CONSIDERATIONS 26

E. FILTER ADAPTATION 28

1. Background 28

2. Innovations Statistics and Maneuver Detection 31

F. ESTIMATION OF PILOT RESPONSE TO TARGET BEARING INPUTS 34

G. FILTER IMPLEMENTATIONS 37

1. Kalman Filter Covariance of Measurement Noise, R(k) 41

2. Filter Initialization 43

IV. PRESENTATION OF RESULTS 45

A. QUALITATIVE OBSERVATIONS 45

B. QUANTITATIVE RESULTS 48

COMPUTER PROGRAM 73

LIST OF REFERENCES 95

INITIAL DISTRIBUTION LIST 96

LIST OF TABLES

I. Release Point Error Table 50
II. Relative Computation and Memory Costs 51

LIST OF FIGURES

2.1. Model Simulation Coordinate Systems 12
2.2. Pilot/Aircraft Controller Configuration 15
2.3. Pilot Induced X-Y Accelerations 16
2.4. Coordinated Turn Free-Body Diagram 17
2.5. Acceleration Probability Density Model 18
2.6. Model Acceleration Correlation Function 19
2.7. GDBS Model Flow Diagram 21
3.1. Dynamic System Model and Discrete Kalman Filter 25
3.2. First Order Kalman Gain Schedule 27
3.3. Second Order Kalman Filter Gain Schedule 28
4.1. ALFBTA(AB) Horizontal Position Trajectory . . . 52
4.2. ALFBTA(AB) True and Estimated Y-Velocity 53
4.3. ALFBTA(AB) Y-Gain Schedule 54
4.4. KALMN1(K10) Horizontal Position Trajectory . . . 55
4.5. KALMN1(K10) True and Estimated Y-Velocity . . . 56
4.6. KALMN1(K10) Y-Gain Schedule 57
4.7. KALMN1(K11) Horizontal Position Trajectory . . . 58
4.8. KALMN1(K11) True and Estimated Y-Velocity 59
4.9. KALMN1(K11) Y-Gain Schedule 60
4.10. KALMN1(K12) Horizontal Position Trajectory . . . 61
4.11. KALMN1(K12) True and Estimated Y-Velocity . . . 62
4.12. KALMN1(K12) Y-Gain Schedule 63
4.13. KALMN2(K20) Horizontal Position Trajectory . . . 64
4.14. KALMN2(K20) True and Estimated Y-Velocity . . . 65
4.15. KALMN2(K20) True and Estimated Y-Acceleration . 66
4.16. KALMN2(21) Horizontal Position Trajectory . . . 67
4.17. KALMN2(K21) True and Estimated Y-Velocity . . . 68
4.18. KALMN2(K21) True and Estimated Y-Acceleration . 69
4.19. KALMN2(K22) Horizontal Position Trajectory . . . 70
4.20. KALMN2(K22) True and Estimated Y-Velocity . . . 71
4.21. KALMN(K22) True and Estimated Y-Acceleration . . 72

ACKNOWLEDGEMENT

I would like to thank my wife, Christy, and my son, Matthew, for sacrificing a great deal of our time together so that this thesis could be realized. Their love, patience, and understanding was more help to me than they could ever imagine. Professor H. A. Titus deserves special recognition for stimulating my interest in the subject of this thesis and then providing encouragement and guidance until its completion.

I. INTRODUCTION

The ground directed bombing system simulated is conceptually similar to the USMC AN/TPB-1D produced by Sierra Research Corporation. This system tracks the tactical aircraft with a conical scan radar, filters the noisy radar data, calculates heading commands based on the smoothed trajectory, and transmits this guidance information to the pilot via the Tacan navigation system located in the cockpit. This heading information directs the pilot to fly the aircraft so that its ground track vector passes through the calculated ordnance release point. Audio signals transmitted to the pilot designate the bomb release time.

In an operational environment such a system would possibly be required to track and guide aircraft conducting significant maneuvers enroute to the target. These maneuvers would most likely be dictated by tactical doctrine or by the threat environment.

With this operational model in mind an appropriate concern is the capability of a ground directed bombing system to track and guide an aircraft exhibiting random maneuvers until moments prior to bomb release. It is obvious that the smoothing filter should be able to respond

to maneuvers, yet settle quickly to an accurate solution as the pilot steadies the aircraft. These conflicting requirements are investigated utilizing both alpha-beta and Kalman filtering techniques.

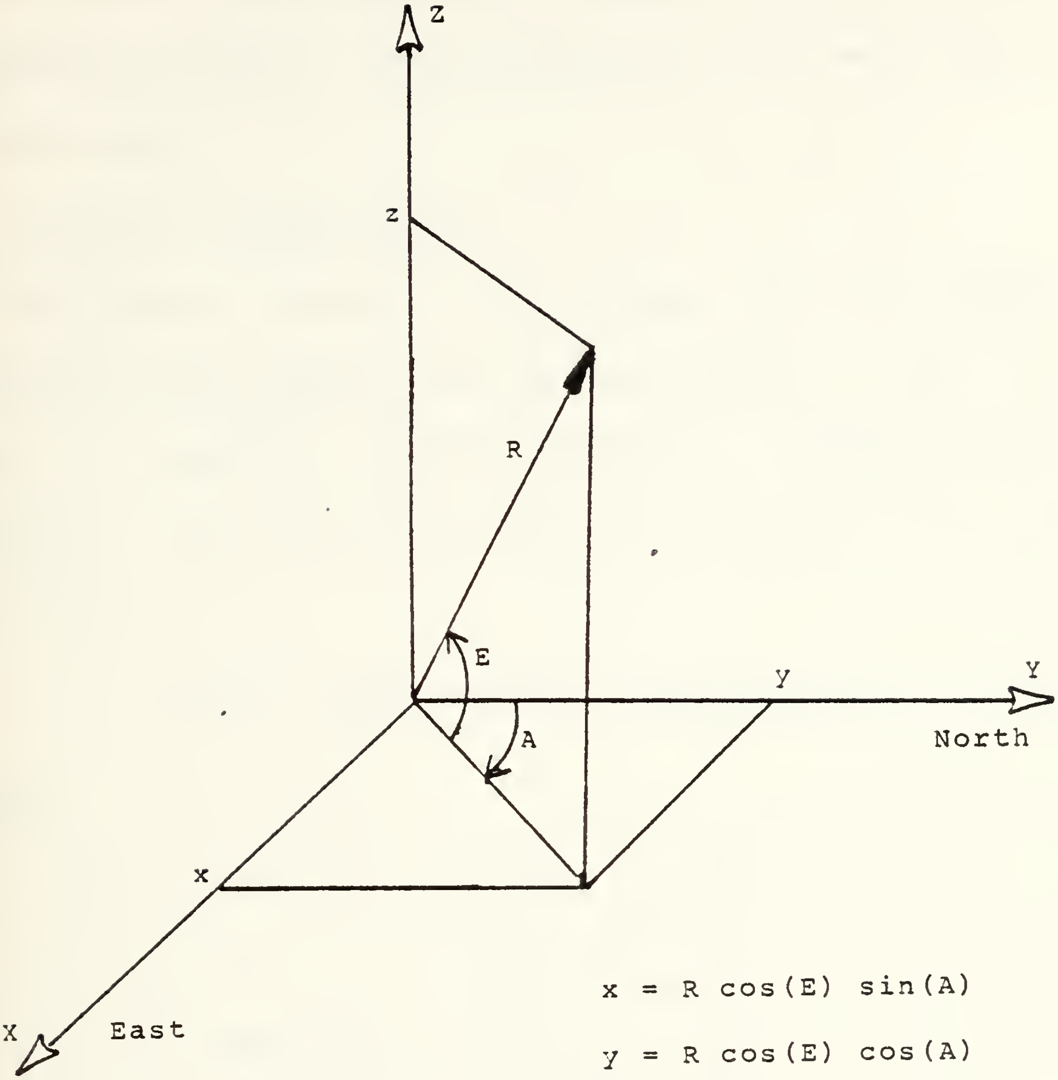
Similar filtering techniques were utilized for a simulation of the USMC AN/TPQ-27, [1]. However in that ground directed bombing system, control signals were directly coupled to the aircraft aerodynamic controls, thus eliminating the uncertainty of pilot response in the control loop. In that study significant improvements in filter response and accuracy were realized by including deterministic forcing autopilot commands in the state estimation via the Kalman filters.

II. GROUND DIRECTED BOMBING SYSTEM (GDBS) SIMULATION

A. COORDINATE SYSTEMS

A Cartesian coordinate system was chosen for the aircraft dynamics model and a radar centered inertial reference frame. In this reference system the y-axis is directed toward true North and the x-axis toward the east. The z-axis is directed away from the center of the earth. All radar measurements of aircraft position, however, are obtained in spherical polar coordinates, i.e. slant range (R), azimith angle from true North (A), and elevation angle (E) from the horizontal. Figure 2.1 shows these coordinate systems and their transformation relationship. Wind is modeled with a constant velocity in the x-y plane with no vertical component.

Curvature of the earth and the fact that pilot heading information is oriented to magnetic north, were not taken into account in the simulation. Also bomb ballistics and therefore coriolis forces were not included in the model. The aircraft is simply directed to a release point in space, which in a full simulation would be derived from the projected bomb trajectory ,ballistic winds, coriolis forces, and a number of other factors, all of which are important to



$$x = R\cos(E)\sin(A)$$

$$y = R\cos(E)\cos(A)$$

$$z = R\sin(E)$$

Fig. 2.1. Model Simulation Coordinate Systems

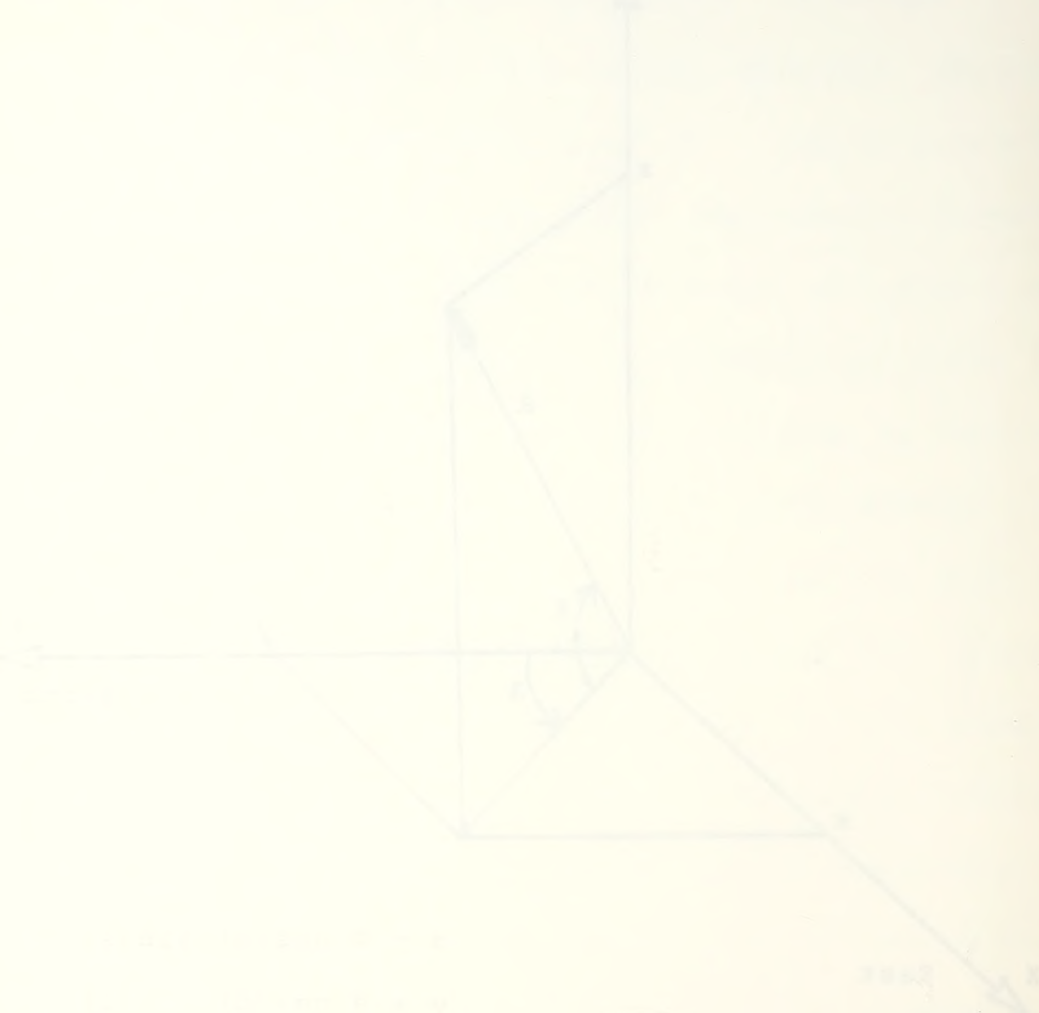


Fig. 2.1. Model Simulation Coordinate Systems

the problem as a whole but are not necessarily germane to the objective of evaluating the response and accuracy of alternative state estimating filters for a goal oriented maneuvering bomber. Thus the simulation has been simplified appropriately.

B. AIRCRAFT DYNAMICS MODEL

The dynamic model of the aircraft for the purpose of simulation was assumed to be a free inertial ($1/s^2$) plant since the bombing profile dictates a constant aircraft airspeed. The discrete realization of this plant is shown in (2.1).

$$X(k+1) = \phi(k+1/k)X(k) + \Delta(k+1/k)U(k) \tag{2.1}$$

where

$$\phi(k+1/k) = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\Delta(k+1/k) = \begin{bmatrix} T^2/2 & 0 & 0 \\ T & 0 & 0 \\ 0 & T^2/2 & 0 \\ 0 & T & 0 \\ 0 & 0 & T^2/2 \\ 0 & 0 & T \end{bmatrix} \qquad (2.3)$$

In the controlled mode the aircraft model simulates the pilot responding to heading inputs transmitted from the radar site to the TACAN navigation system in the cockpit. The model is driven by a pilot/aircraft control function similar to that developed in [1] and shown in Figure 2.2. The input is bearing to the target and the output is a bank angle which generates a heading rate that can be transformed into x-y acceleration components for entry into the dynamic model. It is assumed that in the controlled mode heading changes are made with coordinated turns performed by the pilot in response to heading commands displayed by the TACAN. The pilot/aircraft controller induced x-y accelerations are depicted in Figure 2.3 and summarized by (2.4) and (2.5) below.

$$\ddot{x}(k) = V(k)\cos(\psi(k))\,\dot{\psi}(k) \qquad (2.4)$$

$$\ddot{y}(k) = -V(k)\sin(\psi(k))\,\dot{\psi}(k) \qquad (2.5)$$

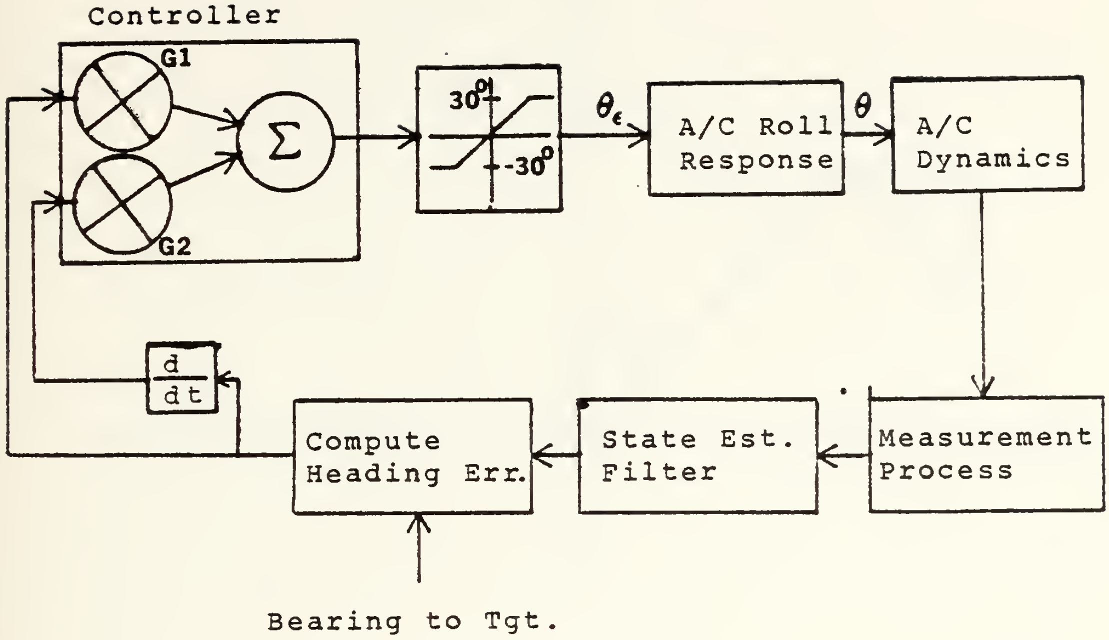


Fig. 2.2. Pilot/Aircraft Controller Configuration

$\psi(k)$ and $V(k)$ are aircraft heading and velocity respectively at time k, and $\dot{\psi}(k)$ is the heading rate which is derived in (2.6) through (2.9) from the free-body diagram shown in Figure 2.4. The aircraft weight is shown in (2.6) below.

$$W = mg = L \cos\theta \qquad (2.6)$$

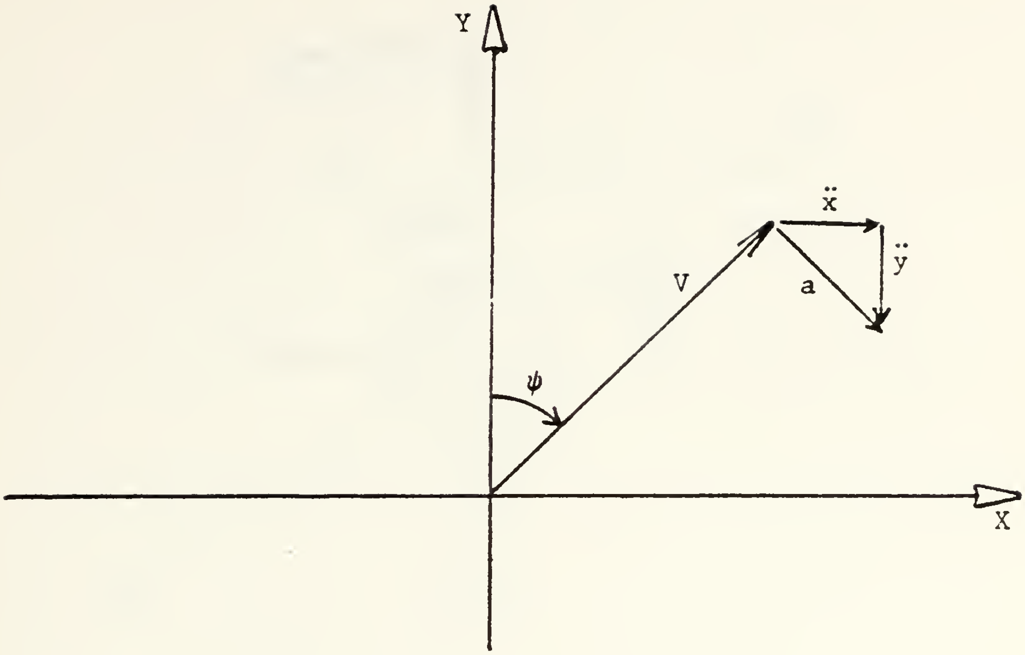


Fig. 2.3. Pilot Induced X-Y Accelerations

Equation (2.7) depicts the centripetal force generated in a turn, where R is the turn radius, L is lift, V is velocity, and θ is the bank angle.

$$P = mV^2/R = L \sin\theta \qquad (2.7)$$

But $V/R = \dot{\psi}$,the turn rate, so

$$P = mV\dot{\psi} = L \sin\theta \qquad (2.8)$$

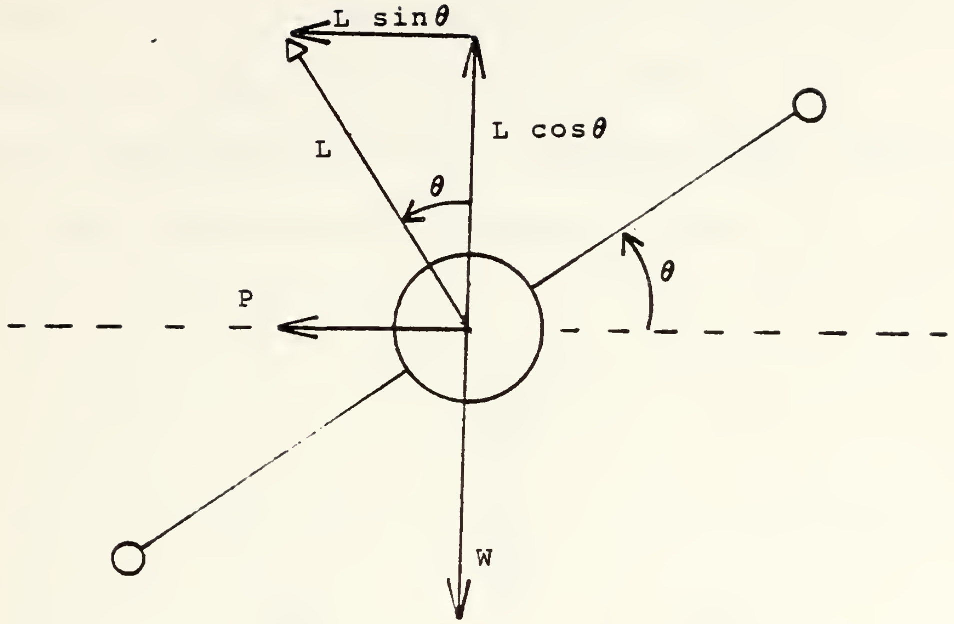


Fig. 2.4. Coordinated Turn Free-Body Diagram

Dividing (2.8) by (2.6) and rearranging terms yields (2.9), which defines $\dot{\psi}$, the aircraft turn rate, as a function of aircraft bank angle θ, and velocity V.

$$\dot{\psi} = g/V \tan\theta \qquad (2.9)$$

From [1] the aircraft roll response is assumed to be of the form shown in (2.10) where τ is the roll response time constant.

$$\frac{\theta}{\theta_\epsilon} = 1/(s\tau + 1) \qquad (2.10)$$

No effort has been made to specifically model pilot delays or response to visual inputs from the TACAN.

In the maneuvering mode the maneuver model described in [2] was used and is shown in Figures 2.5 and 2.6 .

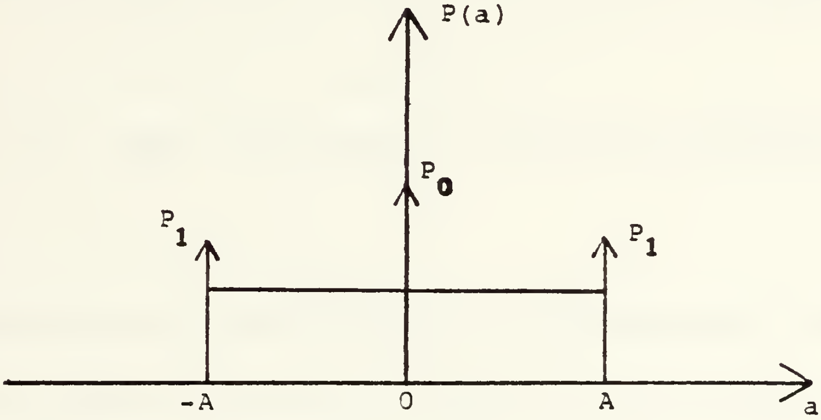


Fig. 2.5. Acceleration Probability Density Model

This was simulated for uncontrolled random flight since the aircraft is assumed to typically move at a constant velocity with turns, evasive maneuvers, and air turbulence interpreted as perturbations upon the constant velocity trajectory. These maneuver perturbations or accelerations

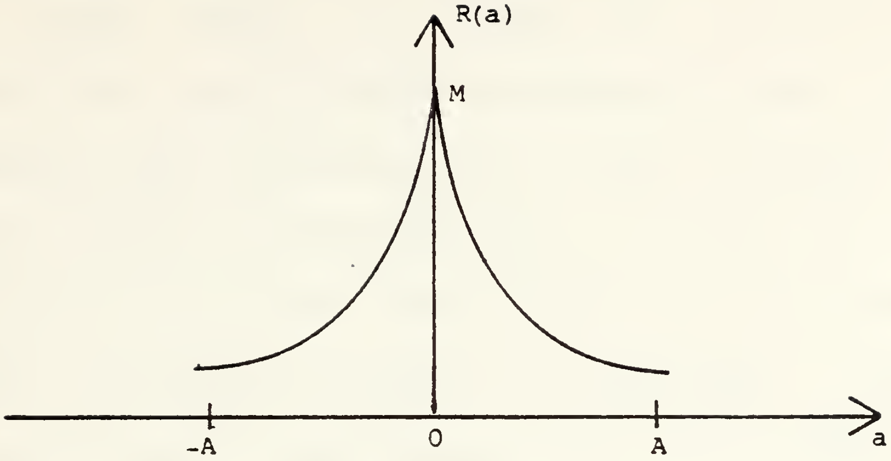


Fig. 2.6. Model Acceleration Correlation Function

can be specified by a magnitude, with probability P(a) from (2.11), and duration of R(a) from (2.12), the correlation function of aircraft acceleration.

$$P(a) = (1-(2P_1 + P_0))/2A \qquad (2.11)$$

$$R(a) = M \exp(-t|a|) \qquad (2.12)$$

The acceleration A in (2.11) is the maximum that can reasonably be expected from the pilot/aircraft in the environment described. P_1 is that probability assigned to the maximum acceleration ±A, P_0 is that probability assigned

to no maneuver, and the assumed probabilitity distribution between these values is uniform with amplitude P(a). Equation (2.12) is the correlation function which yields an acceleration time duration, R(a), which is based on the magnitude of the acceleration |a|. M and t are simply correlation factors which determine how the R(a) varies over the range of possible acceleration amplitudes. From this model one can see how the duration of a high G maneuver for threat avoidance would be considerably less than for a low G clearing turn.

C. GDBS MODEL IMPLEMENTATION

The GDBS model was simulated on an IBM 370 in single precision Fortran. Figure 2.7 shows the basic flow diagram for the computer program, which implements this simulation model. The module labeled 'State Estimation Filter' represents those filter algorithms discussed in the next chapter.

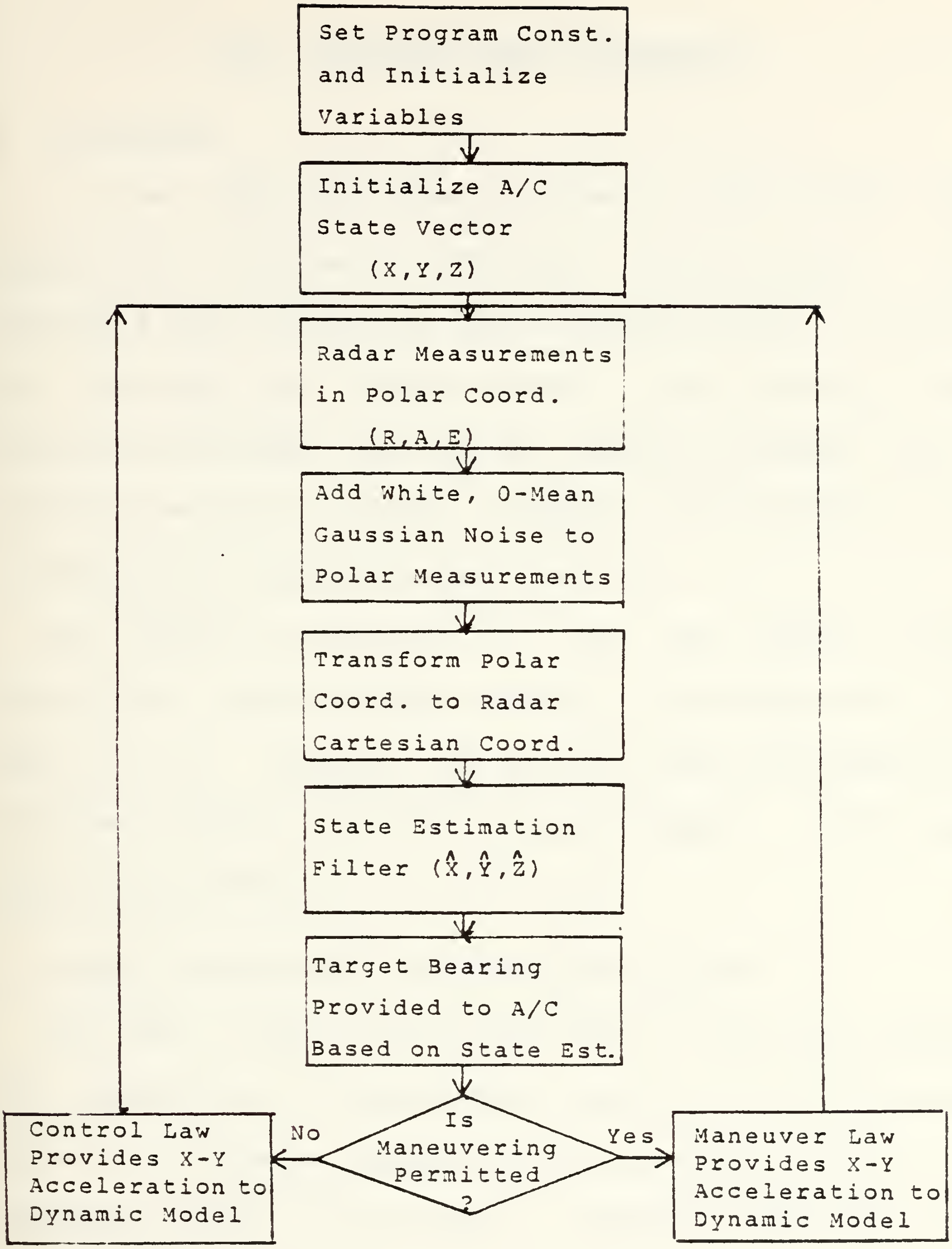

Set Program Const. and Initialize Variables
Initialize A/C State Vector (X,Y,Z)
Radar Measurements in Polar Coord. (R,A,E)
Add White, 0-Mean Gaussian Noise to Polar Measurements
Transform Polar Coord. to Radar Cartesian Coord.
State Estimation Filter ($\hat{X},\hat{Y},\hat{Z}$)
Target Bearing Provided to A/C Based on State Est.
Is Maneuvering Permitted ?
No
Yes
Control Law Provides X-Y Acceleration to Dynamic Model
Maneuver Law Provides X-Y Acceleration to Dynamic Model

Fig. 2.7. GDBS Model Flow Diagram

III. AIRCRAFT STATE ESTIMATION

A. BACKGROUND

A great deal has been written on the theory and application of estimation filters. In particular, [2] provides a good overview of several such filters, including the alpha-beta and Kalman filters, and compares their relative performance not only in terms of accuracy and in response, but also in terms of computer implementation costs in computation time and storage overhead.

The general conclusion is that the Kalman filter out-performs an alpha-beta filter of comparable order by about 2 to 1. However, the cost for such performance is increased computer computation time and memory, of the same relative magnitude.

B. GENERAL DESCRIPTION OF THE ALPHA-BETA FILTER

The basic theory of the alpha-beta filter is derived from minimizing the mean square error of the filtered states. A classic analysis of the alpha-beta filter is provided by [3]. The filter recursive equations are summarized below.

$$x(k/k-1)=x(k-1/k-1)+T\,\dot{x}(k-1/k-1) \qquad (3.1)$$

$$x(k/k) = x(k/k-1) + \alpha(z(k) - x(k/k-1) \tag{3.2}$$

$$\dot{x}(k/k) = \dot{x}(k/k-1) + \beta/T\ (z(k) - x(k/k-1)) \tag{3.3}$$

$x(k/k-1)$ is the predicted position, $X(k/k)$ is the updated position, $\dot{x}(k/k)$ is the updated velocity, and $z(k)$ is the noise contaminated measurement of position at the k-th interval. T is the sample rate of the measurement process, α and β are usually fixed real constants. As pointed out in [4] these alpha-beta equations are analogous to a steady state Kalman filter. For typical parameter values the alpha-beta filter is simply low pass with a heavily damped time response. Thus the filter eliminates not only most high frequency measurement and process noise, but also most maneuver energy from the state estimate.

C. GENERAL DESCRIPTION OF THE KALMAN FILTER

The Kalman filter generates a minimum variance estimate of the plant(aircraft) state vector when the measurement and plant process noise statistics are known and conform to the criteria shown below.

$$E(V(k)V(j)^T) = R(k)\,\delta(k,j) \tag{3.4}$$

$$E(\Delta(W(k)\,W(j)^T)\Delta^T) = Q(k)\,\delta(k,j) \qquad (3.5)$$

$$E(V(k)W(j)^T) = 0 \quad \text{for all } k,j \qquad (3.6)$$

where $$\delta(k,j) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases} \qquad (3.7)$$

A linear time-invariant system is assumed, as in the discrete model representation shown in Figure 3.1. $X(k)$ represents the $(n \times 1)$ state vector, $Z(k)$ the $(m \times 1)$ output vector, $\phi(k)$ the state transition matrix, $H(k)$ the $(m \times n)$ observation matrix, $W(k)$ state excitation or process noise, and $V(k)$ the measurement noise.

The Kalman filter recursion algorithm is summarized below.

$$X(k+1) = \phi(k)X(k) + \Delta(k)W(k) \qquad (3.8)$$

$$Z(k) = H(k)X(k) + V(k) \qquad (3.9)$$

$$\hat{X}(k/k-1) = \phi(k,k-1)\hat{X}(k-1/k-1) + \Delta(k)U(k-1) \qquad (3.10)$$

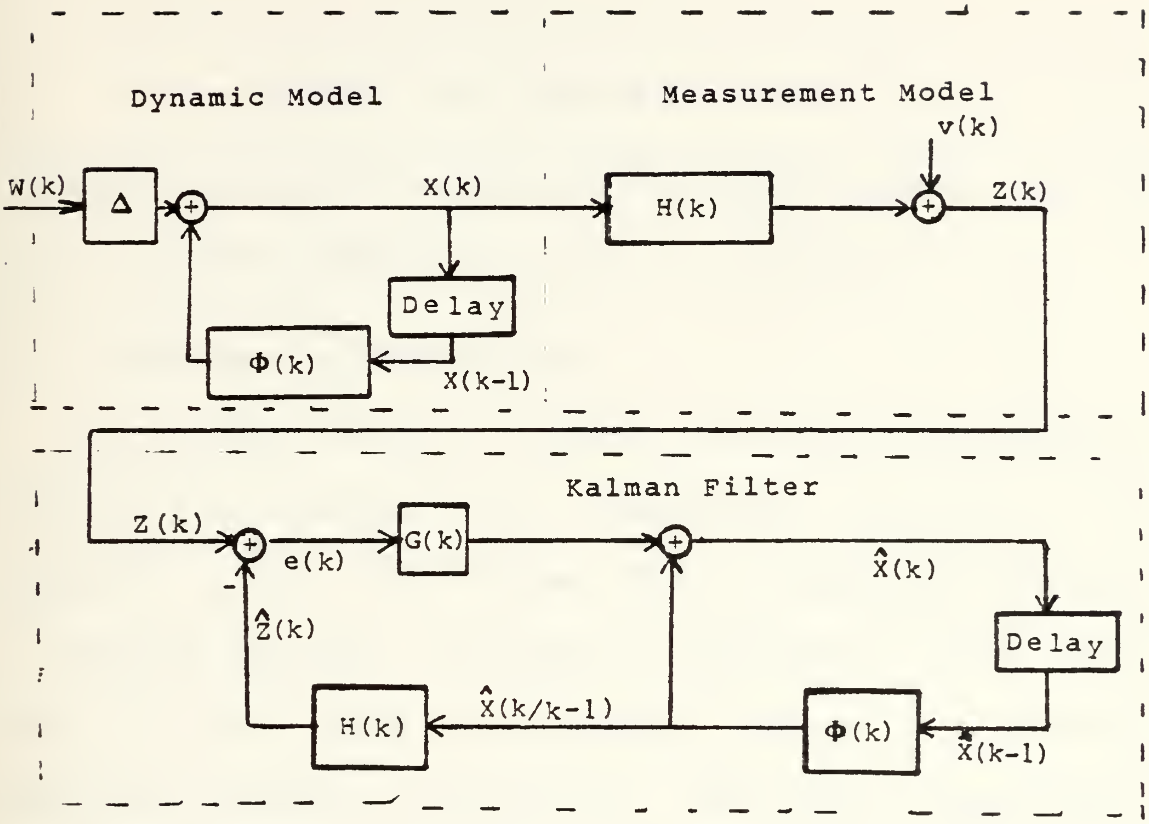


Fig. 3.1. Dynamic System Model and Discrete Kalman Filter

$$P(k/k-1) = \phi(k,k-1)\,P(k-1/k-1)\,\phi(k,k-1)^{T} + Q(k) \tag{3.11}$$

$$G(k) = P(k/k-1)\,H(k)\left[H(k)\,P(k/k-1)\,H(k)^{T} + R(k)\right]^{-1} \tag{3.12}$$

$$\hat{X}(k/k) = \hat{X}(K/K-1)+G(k)\left[Z(k) - H(k)\hat{X}(k/k-1)\right] \quad (3.13)$$

$$P(k/k) = (I - G(k)H(k))P(k/k-1) \quad (3.14)$$

$\hat{X}(k/k-1)$ denotes the estimate of the state vector $X(k)$ based on $(k-1)$ measurements, $Z(1), Z(2), \ldots\ldots Z(k-1)$.

D. FILTER ORDER CONSIDERATIONS

The filter order is chosen to match as closely as possible the expected plant dynamics of the system being modeled. A first order filter would be expected to estimate a constant velocity trajectory effectively and a second order filter would accordingly observe a trajectory exhibiting constant acceleration. The order is used here in the mathematical sense and refers to the order of the differential equation that defines the filter.

Since the aircraft is known to be constrained to a constant velocity profile as it approaches the release point, it would seem reasonable to select a first order filter for modeling. The aircraft dynamics are anticipated to depart from the first order model enroute to the target thus creating transient errors which must be dealt with by filter adaptation. The alternative to this strategy is to

increase the order of the filter to observe these high energy maneuvers for state estimation. However, the settling time of a first order filter is generally less than that of a second order filter as discussed in[5] and graphically illustrated by the first and second order Kalman gain schedules shown in Figures 3.2 and 3.3 respectively.

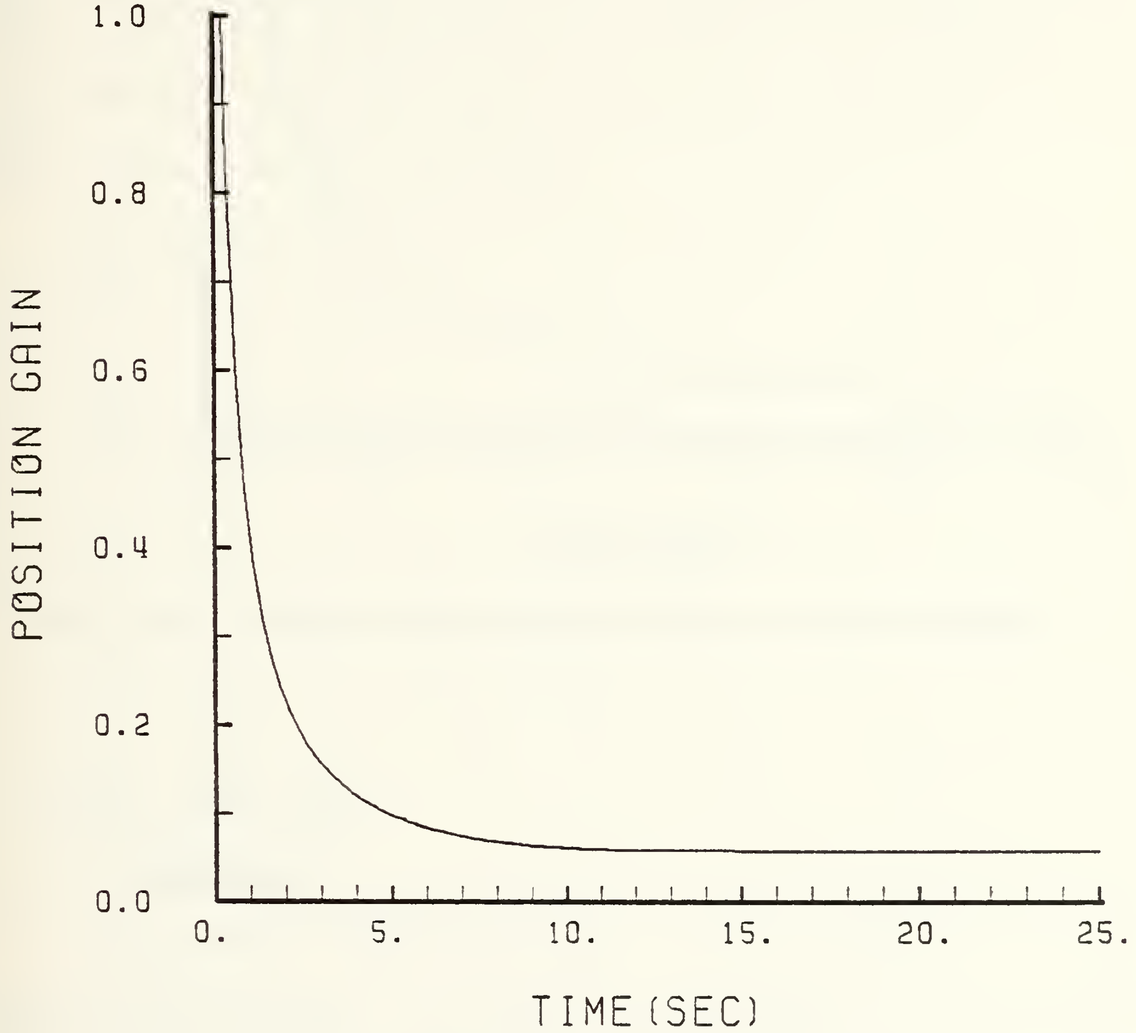


Fig. 3.2. First Order Kalman Gain Schedule

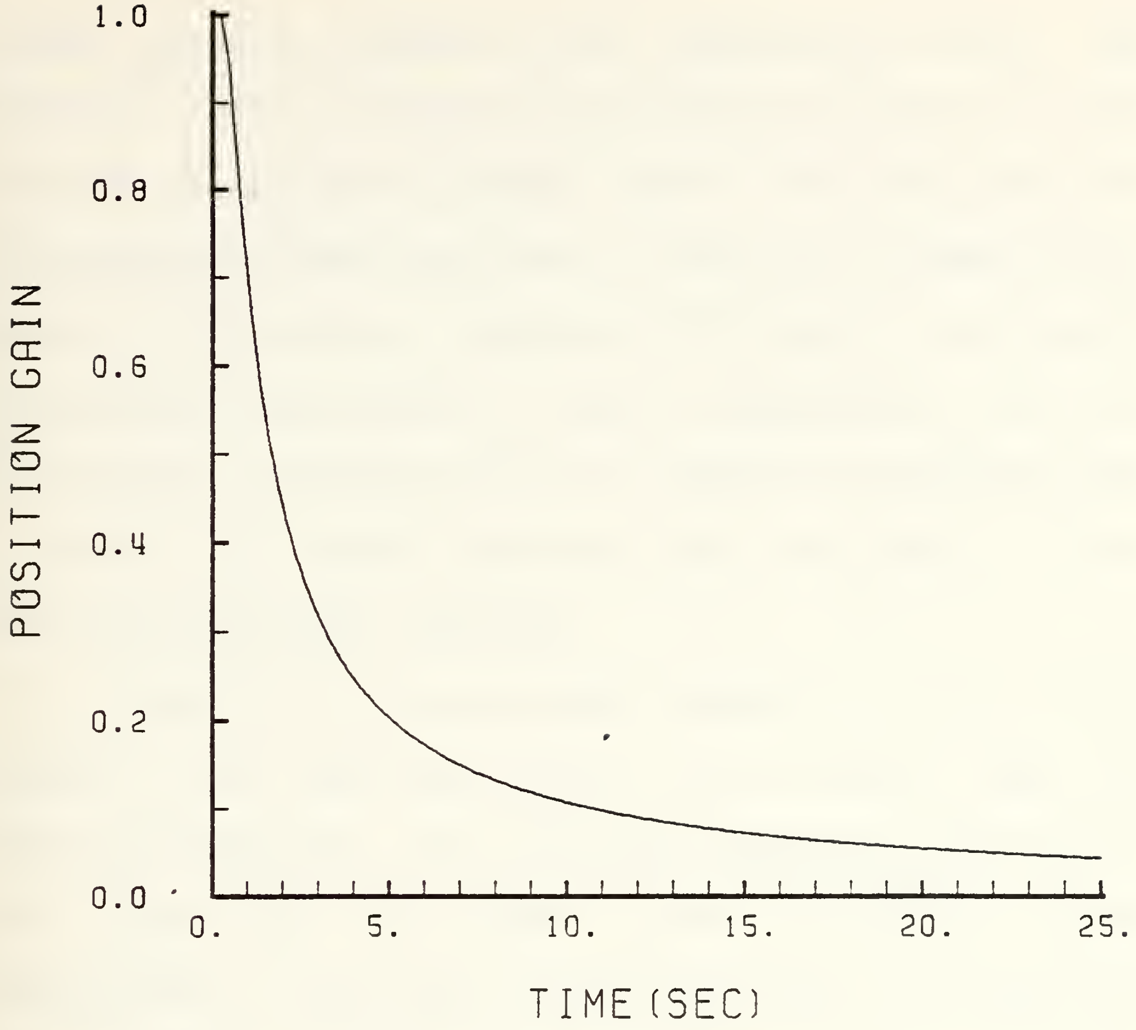


Fig. 3.3. Second Order Kalman Filter Gain Schedule

E. FILTER ADAPTATION

1. Background

No matter what the order or the complexity of the filter type selected, it cannot be expected to fully model the aircraft dynamics and process noise covariance. The model is based on a linear time-invariant system and the

process noise is assumed to be stationary, white, and Gaussian. During portions of the flight, paricularly as the aircraft approaches the release point, the system dynamics are expected to approximate very closely the assumed model. However, during other portions of the flight the aircraft dynamics are anticipated to depart significantly from the filter model. Certainly the pilot should not be constrained to behave in a manner consistent with the model, if the environment dictates otherwise.

Since the pilot/aircraft dynamics are not fully modeled, the suboptimal filter that results might be expected to diverge, e.g. the error covariance generated by the filter and the actual error covariance become inconsistent. The desire is for the filter to transition smoothly between accurate estimations, when the aircraft dynamics conform to those assumed for the model, and less accurate estimations, when the aircraft dynamics do not agree with the model. An adaptive filter realizes this smooth transition by adjusting filter parameters to vary the filter bandwidth to allow a more consistant match between the calculated and actual filter error covariances.

In the case of Kalman filter adaptation, the calculated error covariance becomes a function of the measured data indirectly by making the filter parameters dependent on the observed aircraft motion. The adaptive techniques for the alpha-beta filter are similar in concept. In either case the adaptive process is conceptually straight forward; first divergence is detected, then the filter parameters are modified.

In the case of a ground directed bombing system of the type considered here, the aircraft's behavior could depart from the filter model in a random fashion when the pilot maneuvers in response to a random event in the environment, or deterministically when he responds to target bearing inputs from the ground radar. These two situations may be treated separately or together for the purpose of filter adaptation. To treat them separately, as random and deterministic processes, requires knowledge of the pilot/aircraft response to target bearing inputs. In the case of the ground directed bombing system described in [1] this transfer function was known quite accurately since the input signals from the ground radar were directly coupled to the aircraft aerodynamic controls, with the uncertainties of

pilot response isolated from the control loop. With that information the deterministic forcing could simply be integrated into the filter model. Unfortunately such is not the case for this simulation. A unique approach to this problem will be discussed later. The alternative approach is to consider both processes to be random and proceed from that assumption.

2. Innovations Statistics and Maneuver Detection

As descibed in [5] the innovations or residual sequence of a filter can be observed in order to detect a bias that would indicate divergence of the state estimate from the true state. This is given by

$$\nu(k/k-1) = Z(k) - H(k)\hat{X}(k/k-1) \qquad (3.15)$$

By substituting for Z(k) from (3.9), the measurement model, we see that

$$\nu(k/k-1) = v(k) - H(k)\epsilon(k/k-1) \qquad (3.16)$$

where

$$\epsilon(k/k-1) = \hat{X}(k/k-1) - X(k) \qquad (3.17)$$

Taking expected values , we find that

$$E(\nu(k/k-1)) = 0 \qquad (3.18)$$

$$E(\nu(k/k-1)\,\nu^T(k/k-1)) = R(k) + H(k)P(k/k-1)H^T(k) \qquad (3.19)$$

Thus by referring to the model statistics for R(k) and P(k/k-1), it becomes clear that when the system conforms to the system model, i.e. the filter is operating optimally, the innovation sequence should be zero-mean Gaussian with variance

$$\sigma_\nu^2 = \sigma^2(k) + P(k/k-1) \qquad (3.20)$$

One appraoch to adaptation considers the correlation of the innovation sequence, where the autocovariance

$$\Omega_\nu(i) = E(\nu(k/k-1)\,\nu(k-i/k-1-i)^T) \qquad (3.21)$$

should vanish for $i \neq 0$. Based on these statistics, maneuver detection can be realized by observing the signs of the innovation sequence. The probability that a given sequence is either positive or negative is

$$P(0 > \underset{1-N}{\nu} > 0) = .5^N \qquad (3.22)$$

Another approach utilizes (3.23)

$$\left| \bar{\nu}_N(k) \right| > C\,\sigma_{\bar{\nu}_N}(k) \qquad (3.23)$$

to declare a maneuver when $\bar{\nu}_N$ exceeds a specified value, usually two or three standard deviations of $\sigma_{\bar{\nu}_N}$.

Subsequent to maneuver detection, the filter parameters must be modified to correct filter divergence. Reference [5] summarizes numerous techniques, some being quite complex and computation intensive. The strategy chosen for this simulation was simply to reset the error covariance in response to a detected maneuver. In the situation of a ground directed bombing system, the resulting cost of a false detection becomes high only as the aircraft approaches the release point. This cost can be reduced by disabling filter adaption within a specified time to go.

Still another approach attempts to adapt the filter bandwidth by adjusting Q(k) in (3.11). This approach, investigated in [1] and [6] calculates Q(k) by

$$Q(k) = a\,\mathrm{Del}(k)\mathrm{Del}(k) + b\,\mathrm{Del}(k-1)\mathrm{Del}(k-1) \qquad (3.24)$$

where a and b are determined by data analysis, and

$$\mathrm{Del}(k) = \hat{X}(k/k) - \hat{X}(k/k-1) \qquad (3.25)$$

A variation of this technique that looks at only the change in the highest order state is investigated in the Kalman filter simulation.

Most of the discussion thus far concerning filter adaptation has been directed toward Kalman filters. Most

approaches to adapting alpha-beta filter simply open the bandwith by switching to a different set of parameters when a maneuver has been detected. Reference [4] discusses an adaptation scheme that is more nearly optimal in the sense of covariance matching. However, for the sake of comparison the simple parameter switching technique is implemented in the alpha-beta filter simulation subroutine, since that is the approach used in the AN/TPB-1D.

F. ESTIMATION OF PILOT RESPONSE TO TARGET BEARING INPUTS

As discussed in the previous section, aircraft dynamics depart from the filter model randomly when the pilot responds to events in the environment and deterministically when he responds to target bearing inputs from the GDBS. If his response to these inputs were known with some degree of certainty, then deteministic forcing might be included in the filter model in a manner similar to that found in and [1]. The importance of identifying parameters which define a system so that modern control strategies can be implemented is discussed in [7]. In this case the parameters would be those that describe the pilot/aircraft response to heading inputs.

By using an Autoregressive Moving Average (ARMA) representation for the pilot/aircraft system, and Kalman filtering to process the heading-in(actually heading error from the veiw point of the pilot), bankangle out data, the coefficients associated with the ARMA equation could be identified. From [7] we know that the pilot/aircraft system can be represented by the ARMA equation

$$\theta(k) = \sum_{j=o}^{m} a_j \; \theta_\xi(k-j) - \sum_{j=1}^{n} b_j \; \theta(k-j) \qquad (3.26)$$

where the present bankangle output, $\theta(k)$, is a linear combination of past outputs, $\theta(k-j)$, and of past and present heading error inputs, $\theta_\xi(k)$. Estimating the coefficients of this ARMA equation can be formulated as an adaptive Kalman identifier, where the heading error and the bank angle are simple functions of the velocity and acceleration state estimates generated by the Kalman state estimation filter previously discussed.

If the a_j and b_j coefficients of the ARMA equation are treated as states of the pilot/aircraft system, then the state vector becomes

$$\begin{bmatrix} \underline{a} \\ \hline \underline{b} \end{bmatrix} \qquad (3.27)$$

We assume that these coefficients experience random perturbations so that

$$\begin{aligned} a_i(k+1) &= a_i(k) + w_i(k) \\ b_j(k+1) &= b_j(k) + w_j(k) \end{aligned} \qquad (3.28)$$

Equation (3.26) then becomes

$$\theta(k) = \sum_{j=o}^{m} a_j \ \theta_\xi(k-j) - \sum_{j=1}^{n} b_j \ \theta(k-j) + v(k) \qquad (3.29)$$

where $w_i(k)$, $w_j(k)$, $v(k)$ are noise processes that have the same statistics described for the Kalman filter earlier.

Combining (3.28) and (3.29) we have

$$\left[\begin{array}{c} \underline{a} \ (k+1) \\ \hline \underline{b} \ (k+1) \end{array}\right] = \left[\begin{array}{c} \underline{a} \ (k) \\ \hline \underline{b} \ (k) \end{array}\right] + \underline{w}(k) \qquad (3.30)$$

The measurment vector is defined,

$$H(k) = \Big[\theta_\xi(k) \quad \theta_\xi(k-1) \ \cdots \ \theta_\xi(k-m) \ - \ - \ \theta(k-1) \ \cdots \ - \ \theta(k-n)\Big] \qquad (3.31)$$

From [7] the solution is then formulated as

$$\left[\begin{array}{c} \underline{\hat{a}} \ (k+1/k) \\ \hline \underline{\hat{b}} \ (k+1/k) \end{array}\right] = \Big[I - G(k)H(k)\Big] \left[\begin{array}{c} \underline{\hat{a}} \ (k/k-1) \\ \hline \underline{\hat{b}} \ (k/k-1) \end{array}\right] + G(k) \ \theta(k) \qquad (3.32)$$

where

$$G(k) = P(k/k-1)\,H^T(k)\Big[H(k)P(k/k-1)H^T(k) + R\Big]^{-1} \qquad (3.33)$$

$$P(k/k) = P(k/k-1) - G(k)H(k)P(k/k-1) + Q \qquad (3.34)$$

$$Q = E\left\{\begin{bmatrix} w_i(k) \\ w_j(k) \end{bmatrix} [w_i(k) \;\; w_j(k)]\right\} \qquad (3.35)$$

$$R = E\,[v(k) \;\; v(k)] \qquad (3.36)$$

and

$$P(k/k-1) = E\left\{\left[\begin{bmatrix} \underline{a}(k) \\ \underline{b}(k) \end{bmatrix} - \begin{bmatrix} \hat{\underline{a}}(k/k) \\ \hat{\underline{b}}(k/k) \end{bmatrix}\right]\left[\begin{bmatrix} \underline{a}(k) \\ \underline{b}(k) \end{bmatrix} - \begin{bmatrix} \hat{\underline{a}}(k/k) \\ \hat{\underline{b}}(k/k) \end{bmatrix}\right]^T\right\} \qquad (3.37)$$

Initialization of the states(coefficients) and the error covariance would be similar to that discussed in the next section.

G. FILTER IMPLEMENTATIONS

Three separate filter subroutines were developed to simulate the filtering of raw radar data generated by the ground radar of the bombing system previously described. All three filter configurations are oriented in the three dimensional Cartesian coordinate system described in Chapter 2 since the aircraft dynamics are assumed to be more nearly linear and well behaved than in the polar coordinate system in which the measurements are generated. This disparity

between the measurement reference frame and the model dynamics reference frame results in a basic nonlinearity when transformations are required from one frame to the other. Probably a better coordinate frame for modeling aircraft motion would be one that translates with the aircraft and is oriented along the velocity vector. Such a coordinate system was found to be very awkward and difficult to implement, especially considering the problem of the transformation nonlinearity just mentioned.

The first of these filters, designated ALFBTA, is a simple sixth order alpha-beta filter with the parameter switching adaptation tecnique described in the previous section. Adaption is initiated when a heading rate of 1 degree per second is observed for period of 5 seconds or more. The second and third subroutines implement sixth (KALMN1) and ninth (KALMN2) order Kalman filters respectively. Two separate adaptive techniques, which were described in the previous section, are included with each of these filters. The first of these adaptive algorithms, designated ADPTV1, adjusts the Q(k) matrix from changes computed in the highest order estimate. The second algorithm, designated ADPTV2, simply resets the covariance

of error matrix P(k) when a bias is detected in the innovations sequence for more than one second. As mentioned before, the difference between the sixth and ninth order filters is that the former do not estimate the acceleration states of the aircraft. The cost of this additional information provided by the ninth order filter is more computation time and computer memory.

The aircraft model is formed by defining a three-dimensional Cartesian state vector

$$X_3 = [x \;\; y \;\; z]^T \tag{3.38}$$

where x, y, and z are each one-dimensional two element state vectors(position and velocity) for ALFBTA and KALMN1, and three element state vectors (position, velocity, and acceleration) for KALMN2. The state prediction equations are given by (3.1) for the alpha-beta filter and (3.10) for both Kalman filters. For the Kalman filter configuration

$$\Phi(k) = \begin{bmatrix} \phi & 0 & 0 \\ 0 & \phi & 0 \\ 0 & 0 & \phi \end{bmatrix} \tag{3.39}$$

$$\Delta(k) = \begin{bmatrix} \Delta & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & \Delta \end{bmatrix} \qquad (3.40)$$

where $\phi(k)$ and $\Delta(k)$ are defined by (3.41) and (3.42) for KALMN1 and (3.43) and (3.44) for KALMN2.

$$\phi(k) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \qquad (3.41)$$

$$\Delta(k) = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \qquad (3.42)$$

$$\phi(k) = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \qquad (3.43)$$

$$\Delta(k) = \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} \qquad (3.44)$$

ϕ and Δ are in general functions of k, however for this simulation they are not since a constant data rate is assumed and no extended predictions are required. The U(k) matrix would be utilized to include deterministic forcing in the model, if this information were available as in [1].

However, since an adaptive Kalman identifier was not implemented to estimate the pilot/aircraft response, no attempt was made to include deterministic forcing in the model.

1. Kalman Filter Covariance of Measurement Noise, R(k)

Kalman filter theory assumes linear relationships among measurements and states as can be seen from (3.9). Since aircraft motion is modeled in a Cartesian reference frame and measurements are generated in a polar reference frame, the resulting relationships among the states and measured values are nonlinear, as can be seen from the transformation equations shown below.

$$x = R\cos(E)\sin(A) \tag{3.45}$$

$$y = R\cos(E)\cos(A) \tag{3.46}$$

$$z = R\sin(E) \tag{3.47}$$

Using these polar/Cartesian transformations to nonlinearly combine the polar observations, three-dimensional Cartesian measurements are generated from (3.9) to form (3.48) below.

$$\begin{bmatrix} z_x(k) \\ z_y(k) \\ z_z(k) \end{bmatrix} = \begin{bmatrix} R\ \cos(E)\ \ \sin(A) \\ R\ \cos(E)\ \ \cos(A) \\ R\ \sin(E) \end{bmatrix} + \begin{bmatrix} v_x(k) \\ v_y(k) \\ v_z(k) \end{bmatrix} \qquad (3.48)$$

where the observation matrix for KALMN1 is

$$H(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad (3.49)$$

and for KALMN2 is

$$H(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad (3.50)$$

In order to compute the measurement error variance it is necessary to first linearize the measurement error. Differentiating equation (3.48) with respect to each of the measurement variables yields (3.51), where s and c represent sin and cosine respectively.

$$J(x) = \begin{bmatrix} sAcE & rcAcE & -rsAsE \\ cAcE & -rsAcE & -rcAsE \\ sE & 0 & rcE \end{bmatrix} \qquad (3.51)$$

Thus we find that the linearized Cartesian errors can be expressed as

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = J(x) \begin{bmatrix} v_R \\ v_A \\ v_E \end{bmatrix} \qquad (3.52)$$

Therefore assuming no cross correlation of polar errors, the linearized Cartesian measurement error covariance matrix is

$$R_C(k) = J(x)\ R_P(k)\ J(x)^T \tag{3.53}$$

The diagonal terms of R (k) are

$$R(1,1) = r^2\ (\sigma_E^2 sE^2 sA^2 + \sigma_A^2 cE^2 cA^2) + \sigma_R^2 cE^2 sA^2 \tag{3.54}$$
$$R(2,2) = r^2\ (\sigma_E^2 sE^2 cA^2 + \sigma_A^2 cE^2 sA^2) + \sigma_R^2 cE^2 cA^2 \tag{3.55}$$
$$R(3,3) = r^2\sigma_E^2\ cE^2 + \sigma_R^2\ sE^2 \tag{3.56}$$

The off-diagonal elements are

$$R(1,2) = r^2\sigma_E^2\ sE^2 sA^2 + (\sigma_R^2 - r^2\sigma_A^2)\ cE^2 sAcA \tag{3.57}$$
$$R(1,3) = (\sigma_R^2 - r^2\sigma_E^2)\ sEcEsA \tag{3.58}$$
$$R(2,3) + (\sigma_R^2 - r^2\sigma_E^2)\ sEcEcA \tag{3.59}$$

and due to the symmetry of R (k)

$$R(2,1) = R(1,2) \tag{3.60}$$
$$R(3,1) = R(1,3) \tag{3.61}$$
$$R(3,2) = R(2,3) \tag{3.62}$$

It should be noted that the R(k) matrix is not constant since it depends on range, azimith, and elevation.

2. Filter Initialization

All three filters were initialized with reasonable state values for position, velocity, and acceleration on the the first pass through the filter, since it is assumed that

the GDBS has maintained a good track for sometime prior to the final leg to the target. Consequently the covariance of error for the initial state prediction vector is not set to an arbitrarily large number such as 10^6, as is often done when there is little confidence in the initial state values. Instead 10^3 is used since it is more consistent with the covariance of error in good initial state values. This simulated pass from some other tracking filter to the filter of interest is realistic and reduces the settling time.

Program constants and constant array calculations for the Kalman H, Φ, Δ, and Q matrices are set up on this first iteration. The process noise W is set at an arbitarily small number to ensure that the gain matrix will not converge to zero and accentuate divergence problems.

IV. PRESENTATION OF RESULTS

A. QUALITATIVE OBSERVATIONS

Seven different filter configurations were ultimately implemented and evaluated with the GDBS simulation. These configurations are discussed below and assigned alpha-numeric symbols for ease of reference.

The sixth order alpha-beta filter, ALFBTA(AB) was implemented with the simple parameter switching technique outlined in Chapter 3. However, the heading rate maneuver detection process proved to be too insensitive to slow maneuvers and was therefore augmented with a trigger that reset filter parameters when the heading error exceeded 3 degrees. The results of this change proved to be worthwhile, as will be shown later.

The next three filters are variations of the sixth order KALMN1 filter. (K10) is KALMN1 without adaptation, (K11) is KALMN1 with the process noise adaptation scheme for modifing Q(k), and (K12) is KALMN1 adapted by resetting P(k), the error covariance matrix, as discussed in Chapter 3. Filters (K20), (K21), and (K22) are variations of KALMN2 which correspond to the KALMN1 variants described above.

Since the objective was to evaluate the accuracy and response of each filter in the final phase of bomb delivery, the simulation was initialized with the aircraft within 90 seconds of the release point, at a speed of 480 knots, and on a heading within 10 degrees of the target bearing. For each run, after allowing 5 seconds for the filter to settle, Tacan target bearing information was provided to the pilot controller. At 15 seconds elapsed time, filter adaptation was enabled and random maneuvers were begun at 20 seconds. Five separate runs were evaluated for each filter where the random maneuvers were ceased at 80, 60, 50, 40, and 30 seconds prior to arrival at the release point.

The following plots were generated for the case where maneuvers were stopped with a time-to-go of 50 seconds. Figures 4.1 through 4.21 show true and estimated(connected symbols) position, velocity, and acceleration trajectories as functions of time for all filter configurations. Note that the target position is designated by a circle on the horizontal trajectory plot. Also representative filter gain schedules are included to show the effects of the particular adaptation process being utilized.

Figures 4.1 and 4.2 indicate the magnitude of the maneuver encountered and the good state estimation qualities of this simple sixth order alpha-beta filter. Notice there is no significant divergence of the estimated trajectory from the true trajectory throughout the run. Figure 4.3 shows the step gain adaptation at the beginning of the run in response to the controlled turn to the target heading. This gain is then reduced after the turn is completed and before the first random maneuver begins, when the gain is again increased.

The trajectory shown in Figure 4.4 contasts sharply with that shown in 4.1, showing significant filter divergence for the nonadaptive sixth order Kalman filter. The significant lag in velocity state estimation shown in Figure 4.5 results from the convergent gain properties characterized in Figure 4.6.

The performance of K10 changes dramatically when it is made adaptive as shown in Figures 4.7, 4.8, and 4.9 for K11 and Figures 4.10, 4.11, and 4.12 for K12. Figure 4.9 shows continuous gain adjustment in response to perceived changes in the process noise. Figure 4.12 shows the effect of resetting the covariance of error in response to a maneuver.

Figures 4.13, and 4.14 show the improvement in state estimation of the K20 nonadaptive Kalman filter when the order is increased over that of K10. It is interesting to note that we see significant overshoot in the velocity estimate for the first time. Unlike the filters discussed thus far, K20 provides an acceleration estimate which can be seen in Figure 4.15 and accounts for the sensitivity of the velocity estimate.

Adapting K20 through the noise process technique results in K21 which produces the position, velocity, and acceleration estimates shown in Figures 4.16, 4.17 and 4.18 respectively. Figures 4.19, 4.20, and 4.21 provide the same information for K22, which represents the covariance of error adaptation variant of K20. These last two adaptive filters show little, if any, apparent improvement over the nonadaptive version. This observation is supported in the following section.

B. QUANTITATIVE RESULTS

A single run, for each filter configuration evaluated for each maneuver termination time, is not sufficient to properly determine filter performance over the range of possible maneuver trajectories and measurement noise

sequences. Therefore, 30 simulation trajectories per filter, per maneuver period, were conducted with different random maneuver and measurement noises sequences generated for each run. The bomb release signal to the pilot was assumed to occur at the closest point of approach(CPA) to the target release point. The average of the resulting CPA's for each 30 test runs are shown in Table I. CPA's greater than 250 feet are classified unsatisfactory and labled 'U' appropriately.

At a glance it is apparent that the adaptive sixth order alpha-beta filter performs very well in such a dynamic environment, except when maneuvers are continued very close to the target. The nonadaptive sixth order Kalman filter is obviously unsuited by itself, but when made adaptive, preforms very well, particularly for the process noise adaptation technique when maneuvers are terminated late in the target run. The ninth order Kalman filter performance, both adaptive and nonadaptive, is comparable to the alpha-beta and adaptive sixth order Kalman filters, but has problems in close due to its longer settling time. Notice that the adaptive variants of the ninth order Kalman have little effect on that filter's performance, as we surmised in the last section.

TABLE I

Release Point Error Table

Filter / TTG (sec)	Alpha-Beta	Kalman (sixth order)			Kalman (ninth order)		
	Adptv	Non-Adptv	Adptv		Non-Adptv	Adptv	
	AB	K10	K11	K12	K20	K21	K22
80	20'	46'	38'	39'	30'	30'	30'
60	43'	U	42'	37'	43'	39'	35'
50	41'	U	75'	53'	50'	65'	52'
40	55'	U	54'	82'	119'	98'	135'
30	230'	U	109'	U	247'	U	U

Table II shows the relative cost, in computation time and memory, to implement KALMN1 and KALMN2 in relation to ALFBTA.

TABLE II

Relative Computation and Memory Costs

Alpha-Beta (sixth ord)	1
Kalman (sixth ord)	2.9
Kalman (ninth ord)	3.4

The only advantage to implementing the most responsive variant of KALMN1 would be to reduce the probability of a GDBS generated abort, due to large predicted bomb impact errors, when maneuvers are carried very close to the release point. Lastly, if the adaptive Kalman identifier proved to be useful in providing deterministic forcing for the Kalman filter model and resulted in improved accuracy and response over the alternatives presented, the cost in computation and memory resources would be even greater than we have seen here.

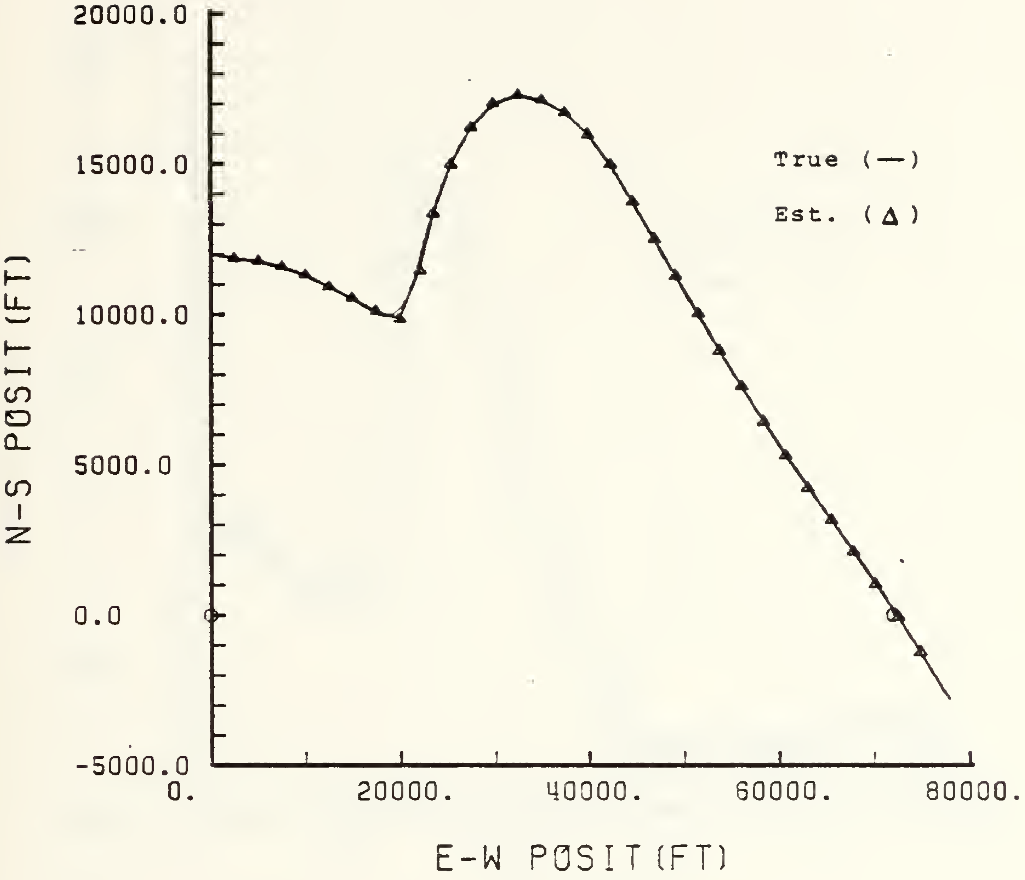


Fig. 4.1. ALFBTA(AB) Horizontal Position Trajectory

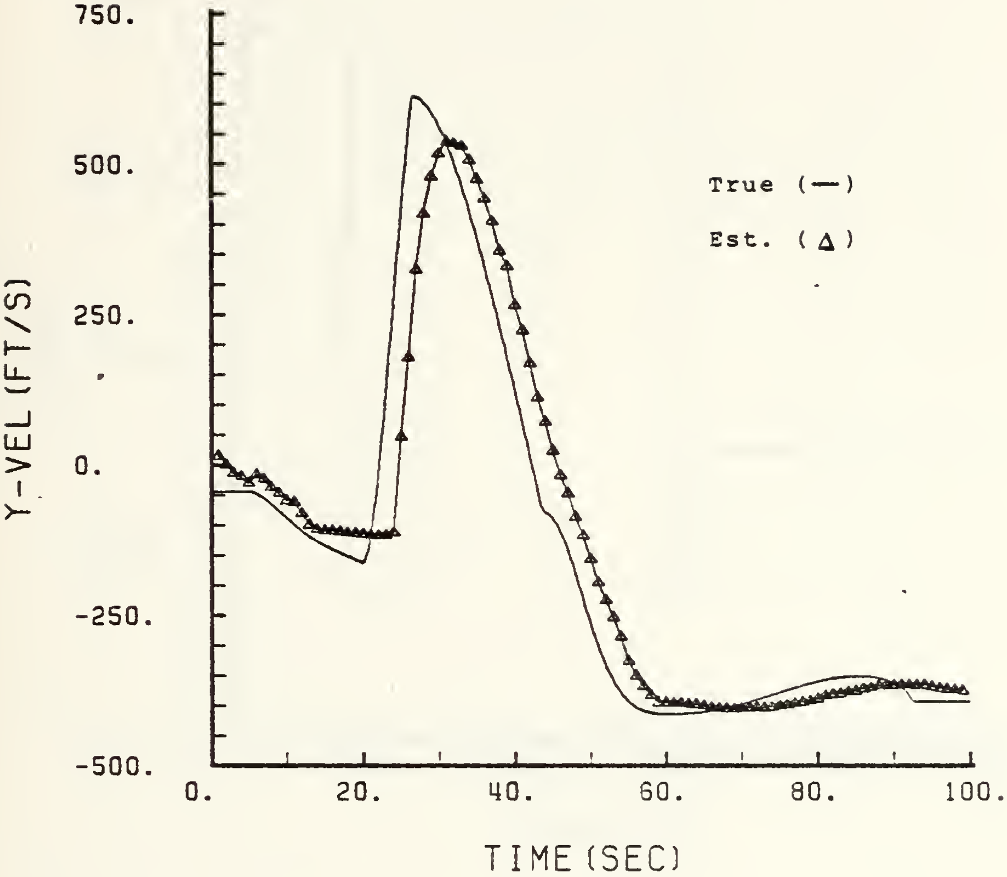


Fig. 4.2. ALFBTA(AB) True and Estimated Y-Velocity

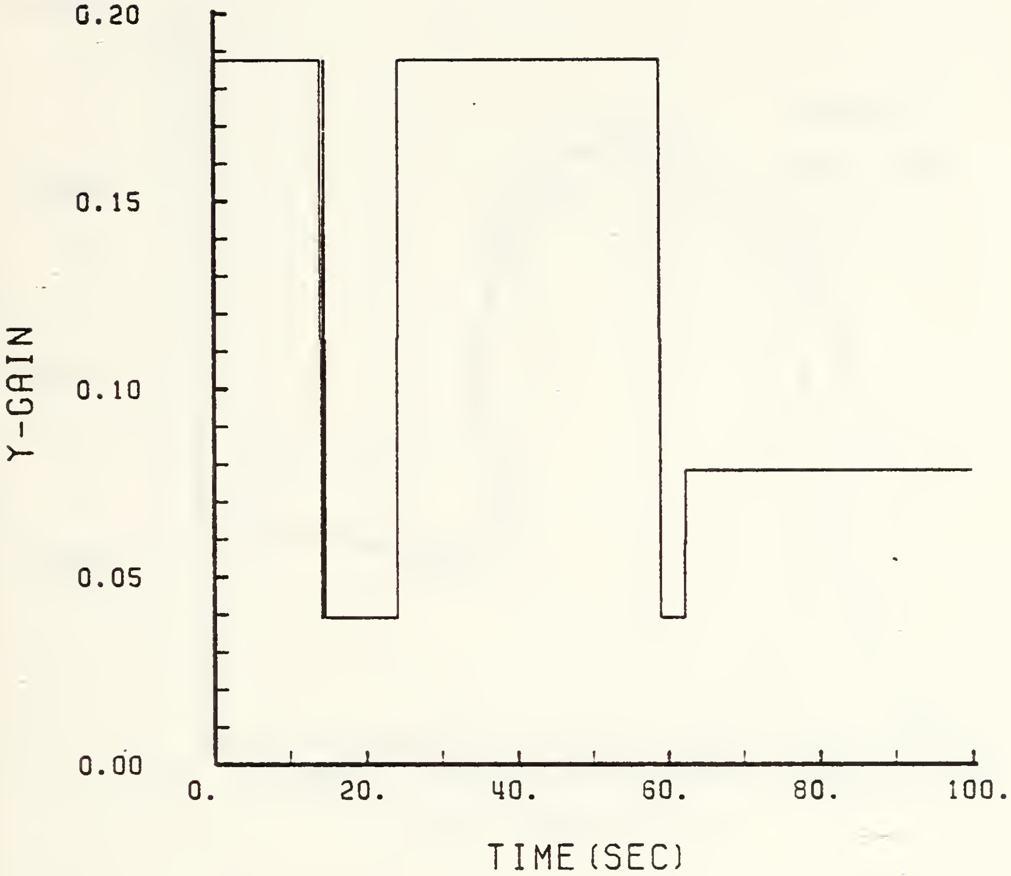


Fig. 4.3. ALFBTA (AB) Y-Gain Schedule

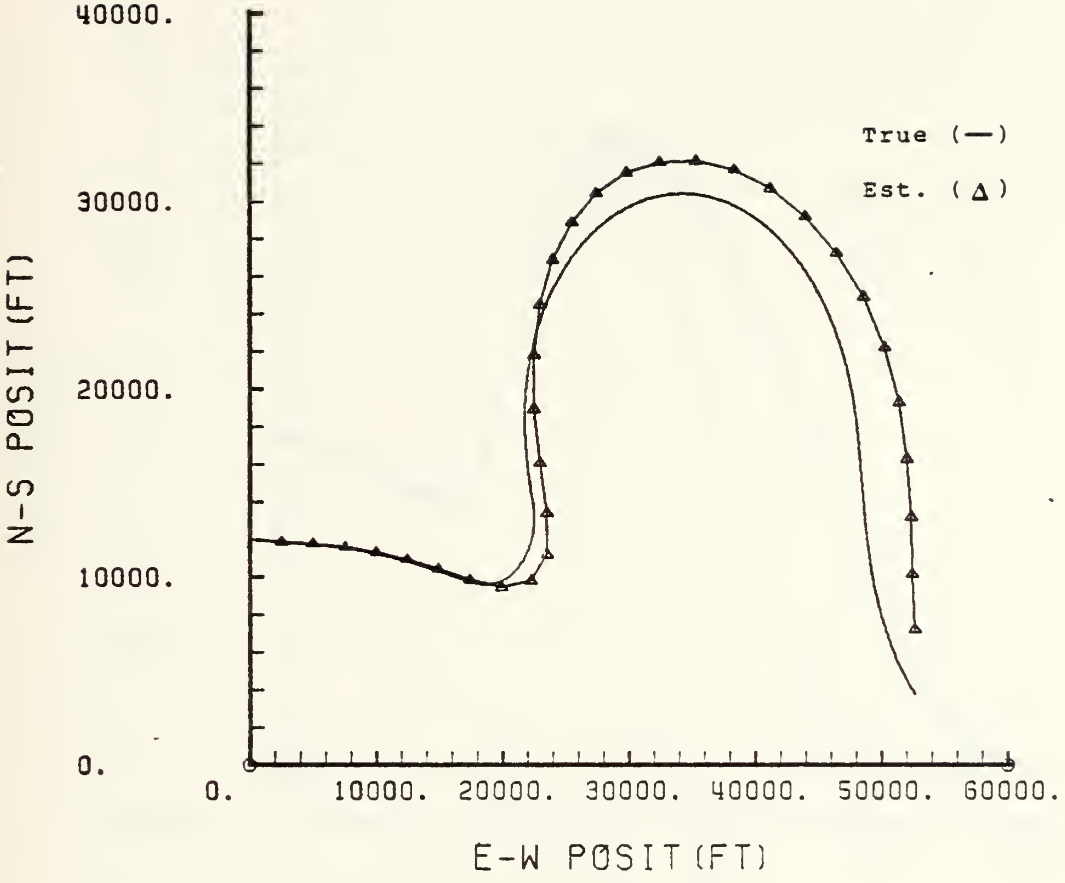


Fig. 4.4. KALMN1(K10) Horizontal Position Trajectory

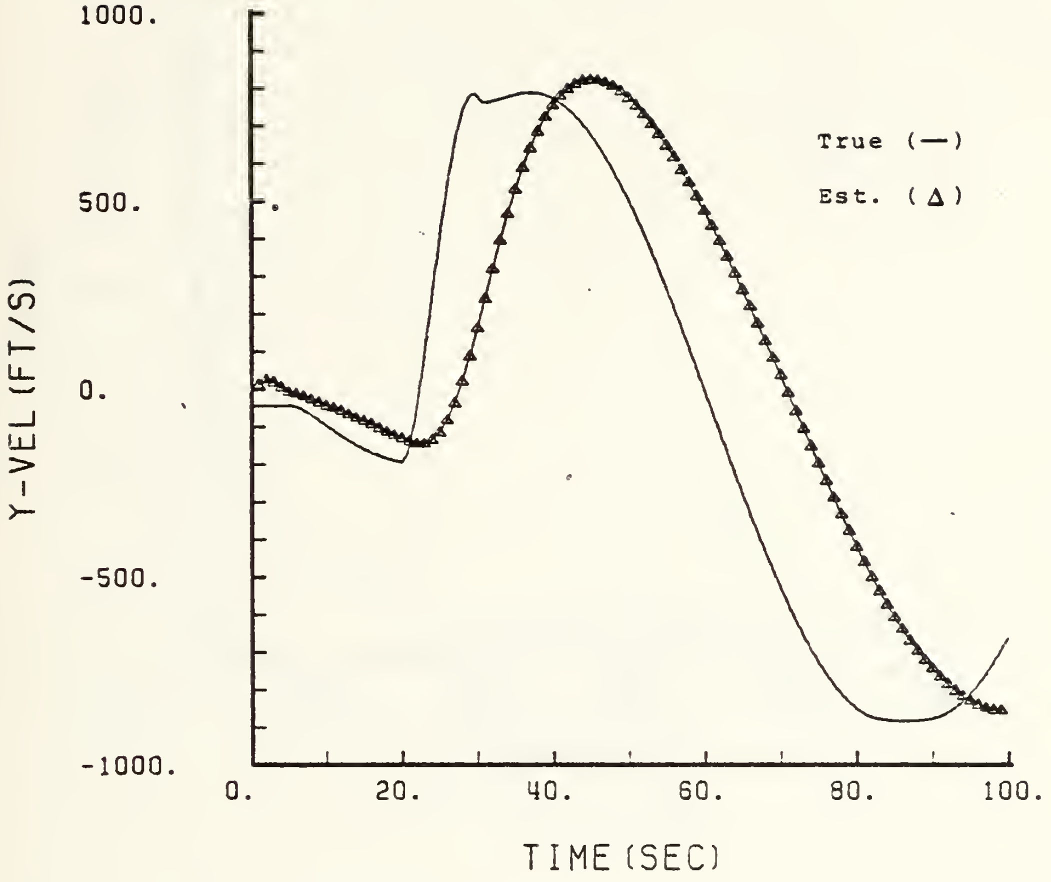


Fig. 4.5. KALMN1(K10) True and Estimated Y-Velocity

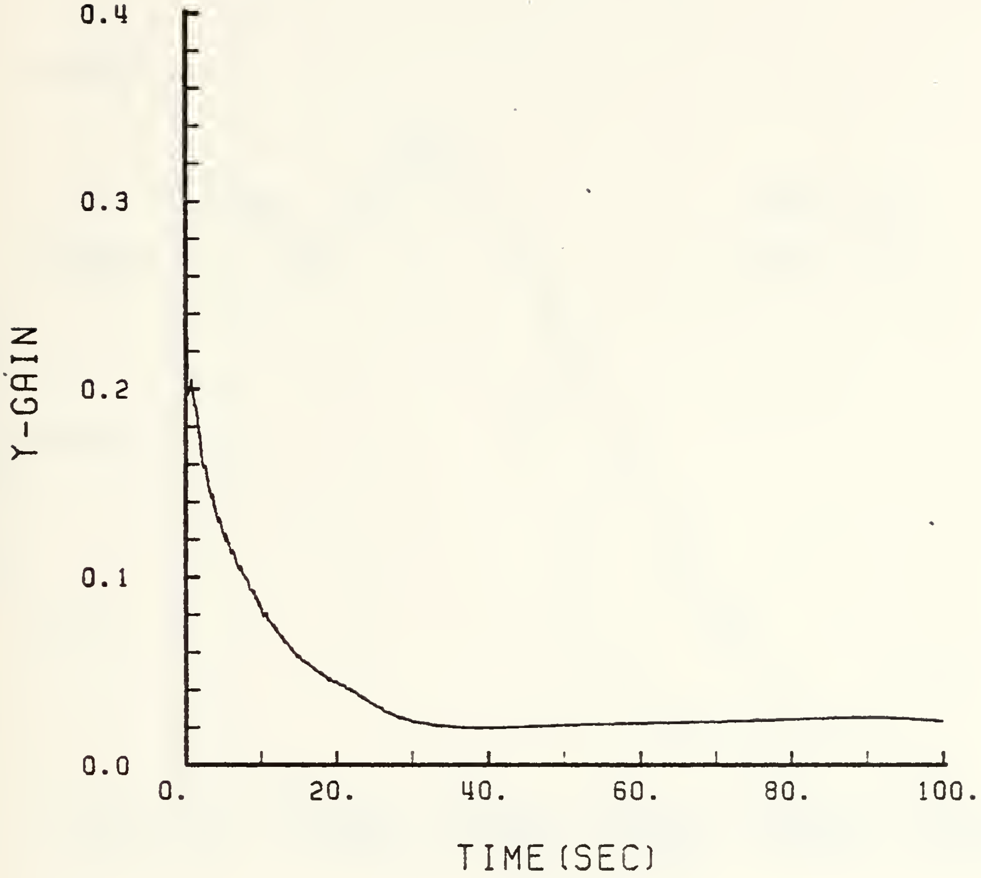


Fig. 4.6. KALMN1(K10) Y-Gain Schedule

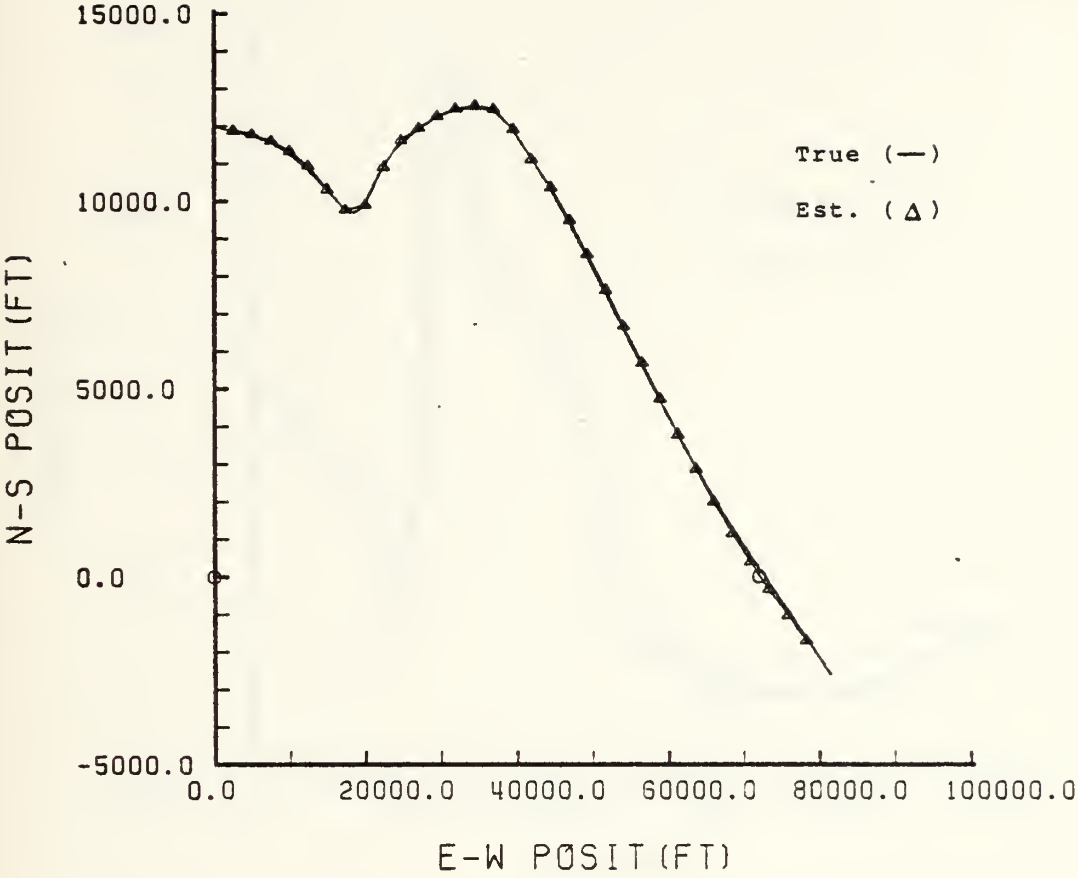


Fig. 4.7. KALMN1(K11) Horizontal Position Trajectory

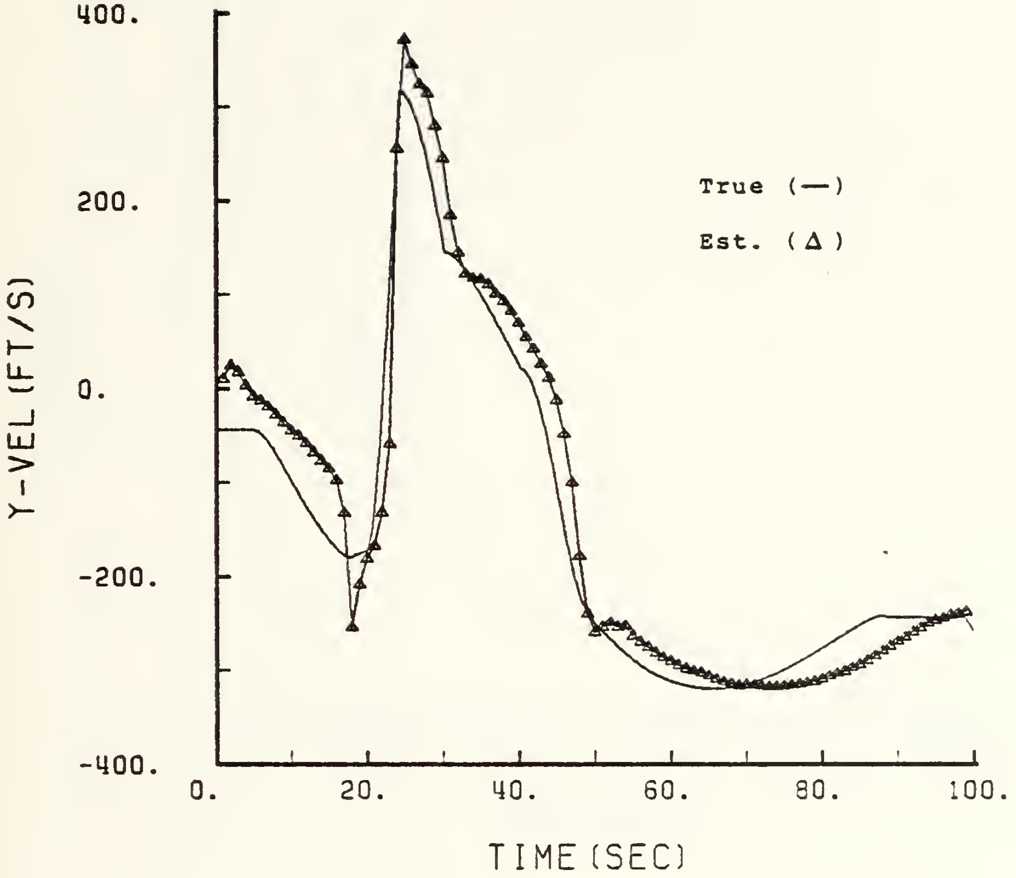


Fig. 4.8. KALMN1(K11) True and Estimated Y-Velocity

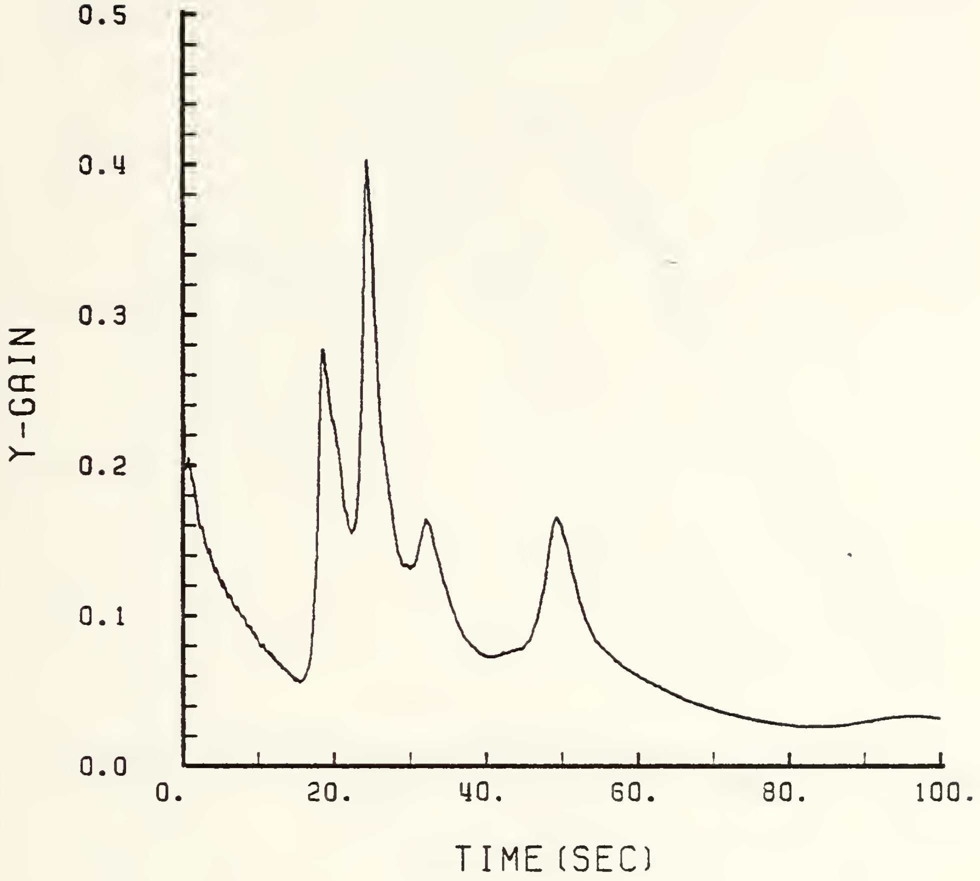


Fig. 4.9. KALMN1(K11) Y-Gain Schedule

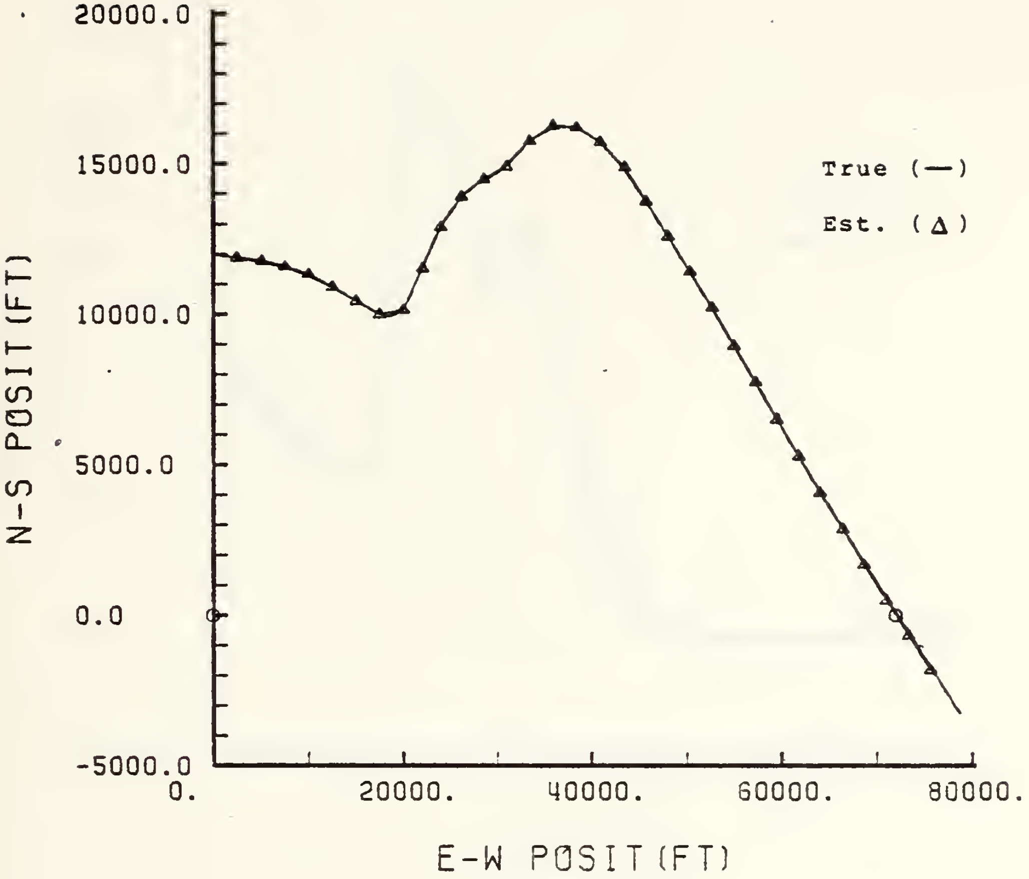


Fig. 4.10. KALMN1(K12) Horizontal Position Trajectory

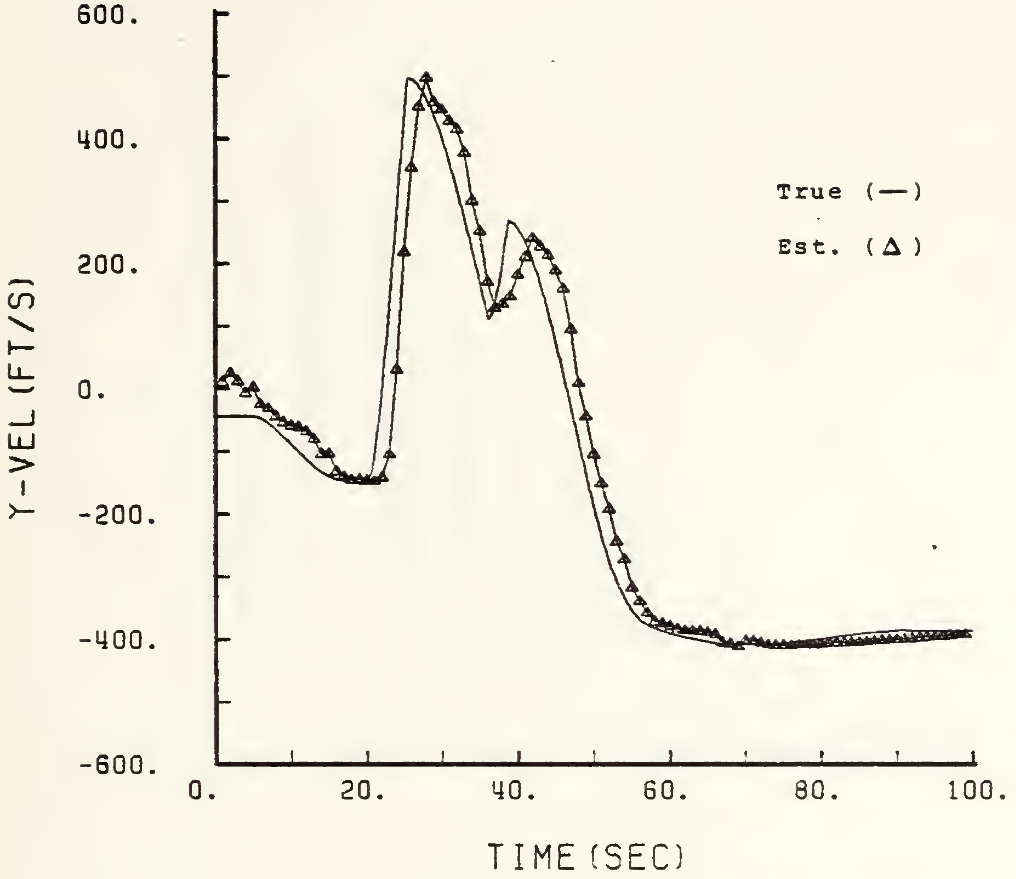


Fig. 4.11. KALMN1(K12) True and Estimated Y-Velocity

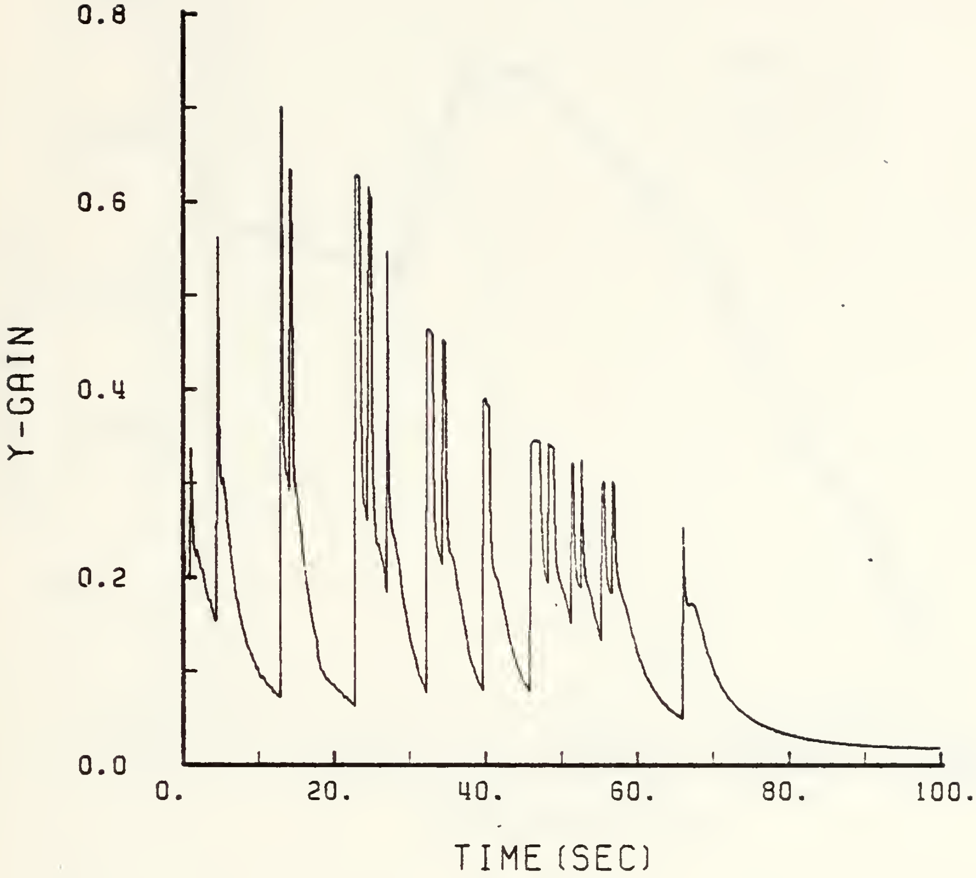


Fig. 4.12. KALMN1(K12) Y-Gain Schedule

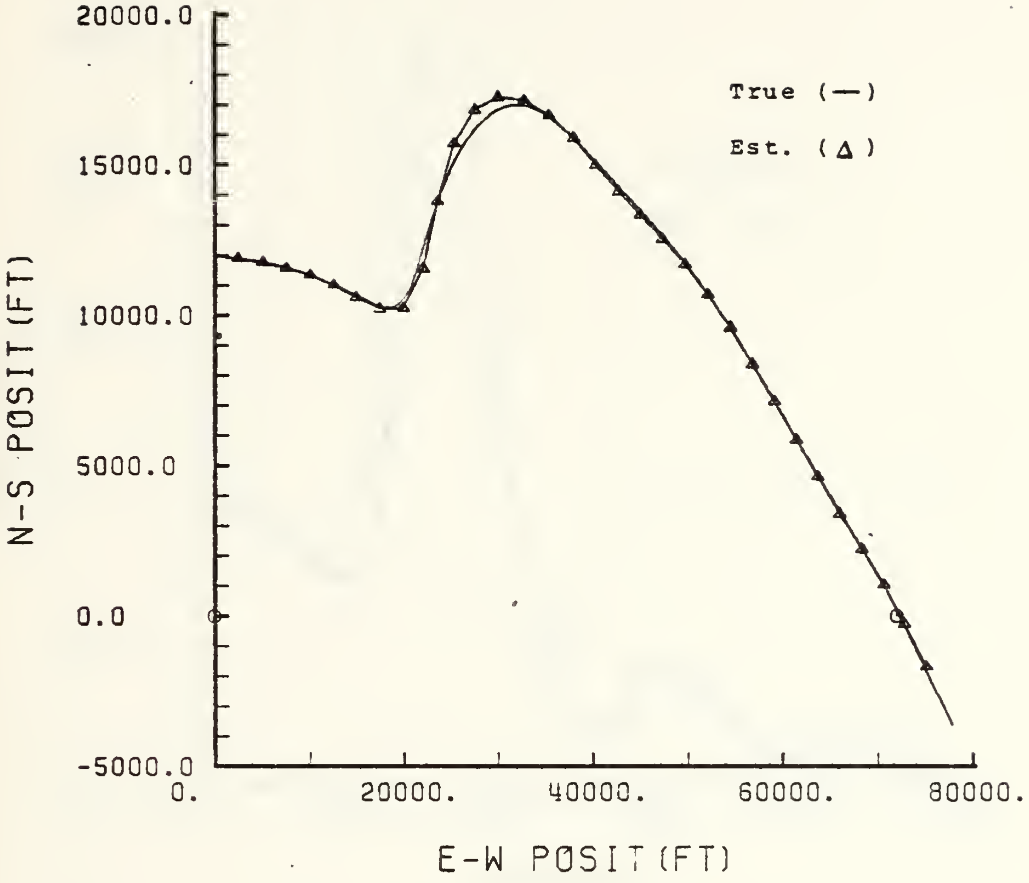


Fig. 4.13. KALMN2(K20) Horizontal Position Trajectory

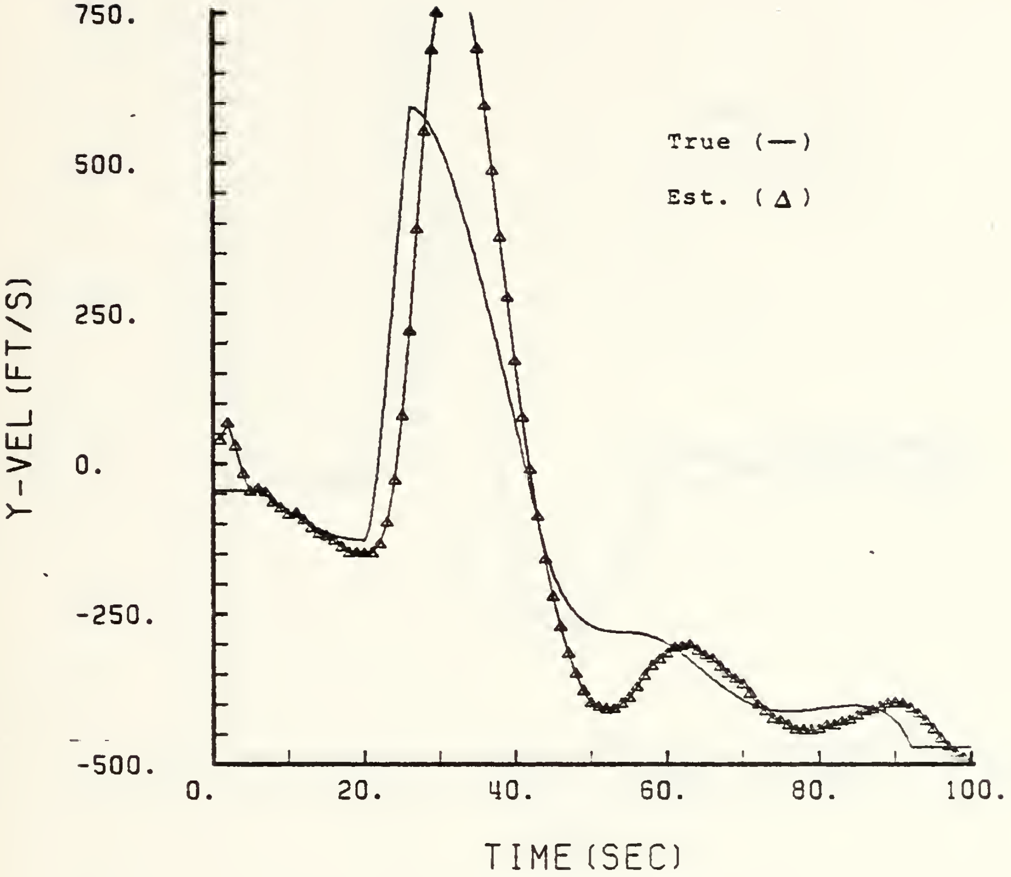


Fig. 4.14. KALMN2(K20) True and Estimated Y-Velocity

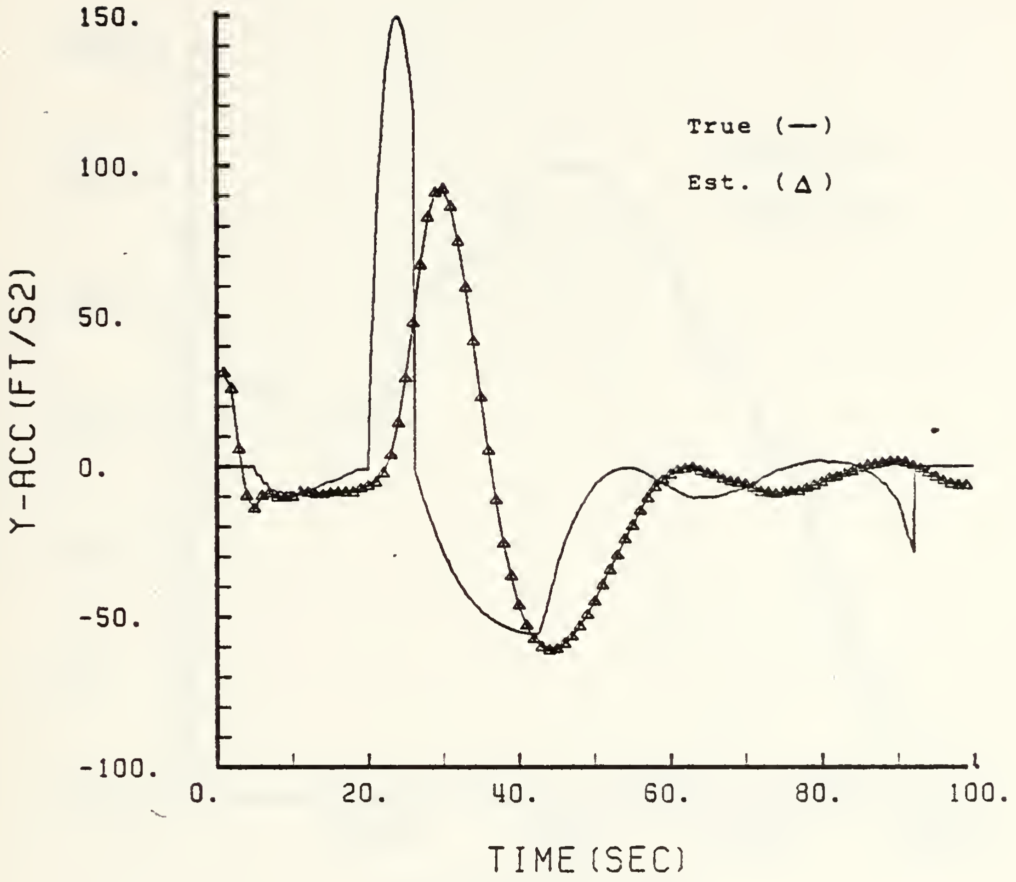


Fig. 4.15. KALMN2 (K20) True and Estimated Y-Acceleration

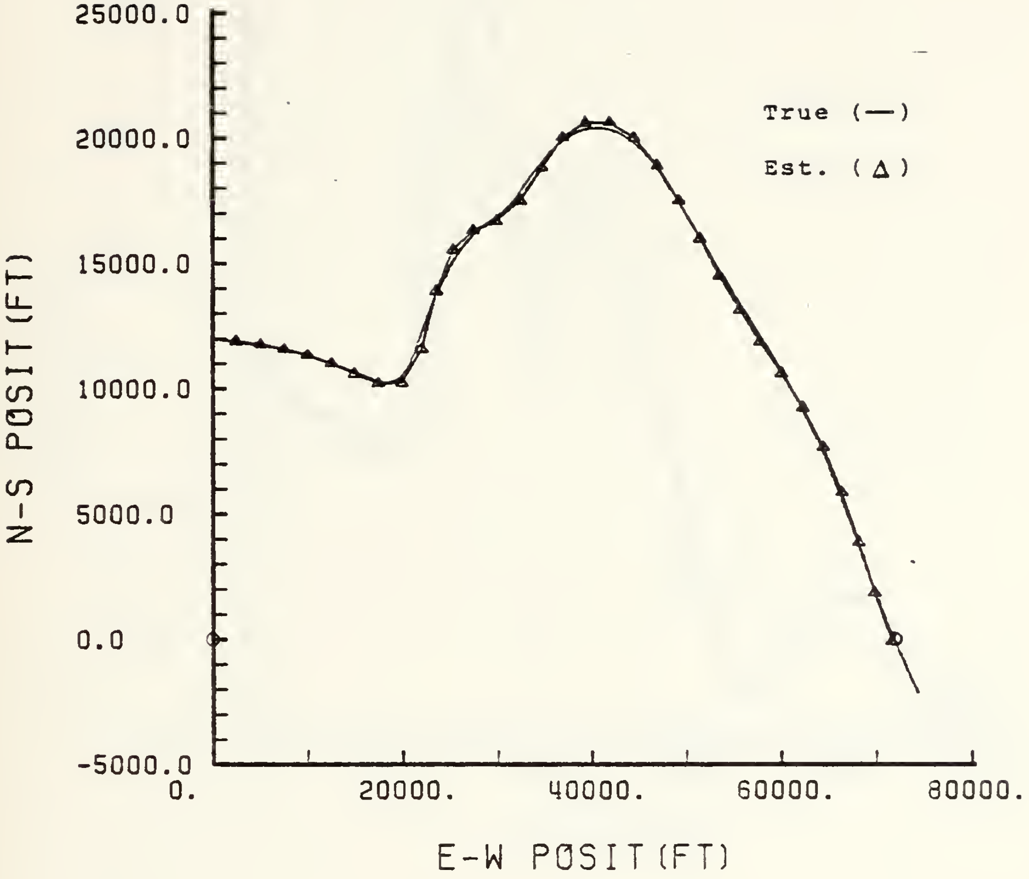


Fig. 4.16. KALMN2(21) Horizontal Position Trajectory

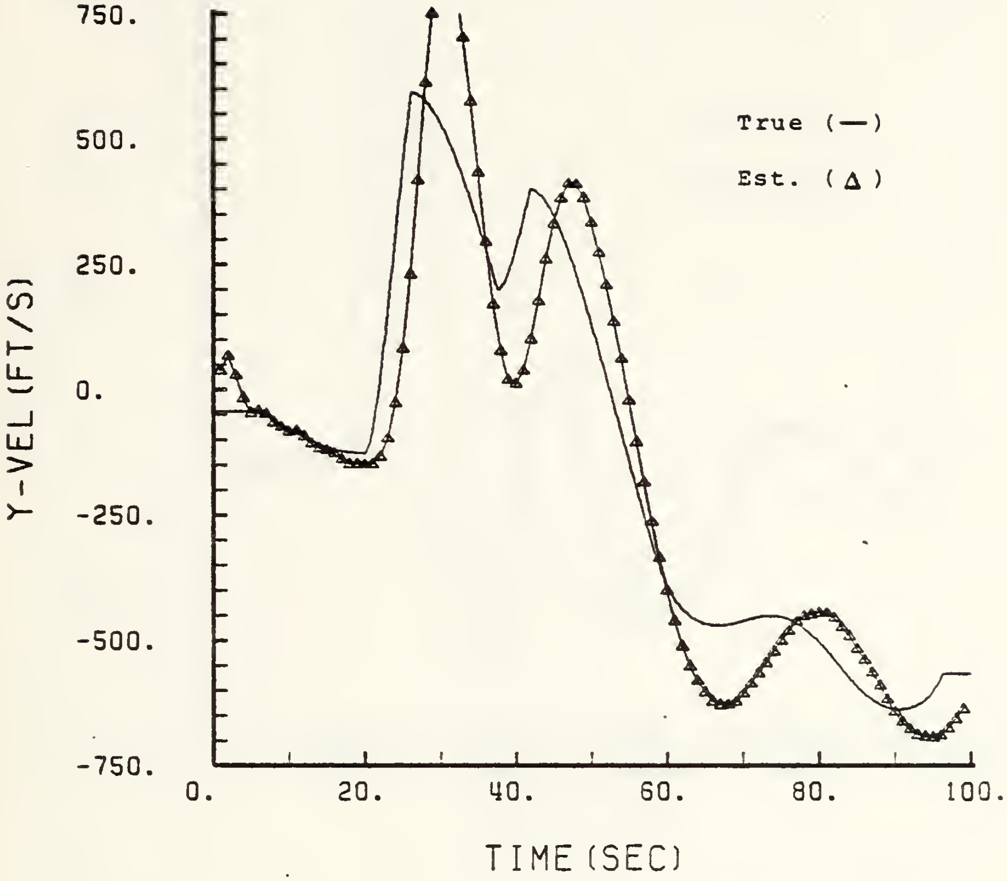


Fig. 4.17. KALMN2(K21) True and Estimated Y-Velocity

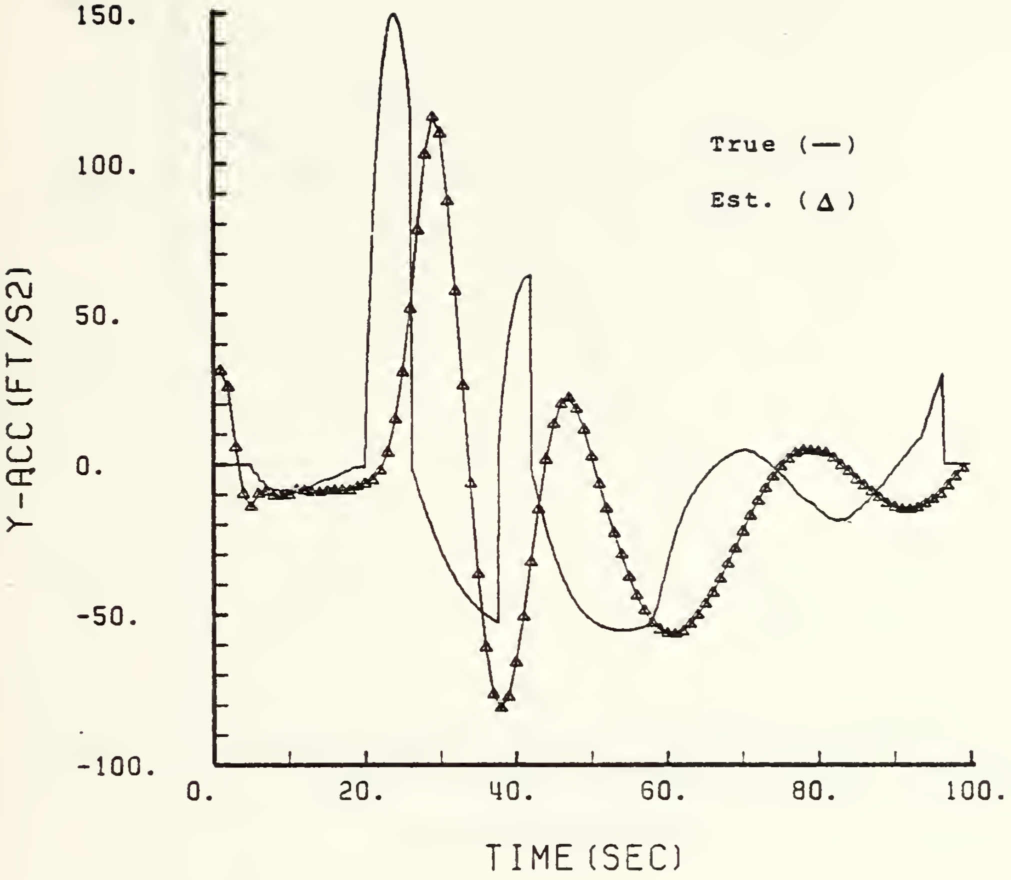


Fig. 4.18. KALMN2 (K2 1) True and Estimated Y-Acceleration

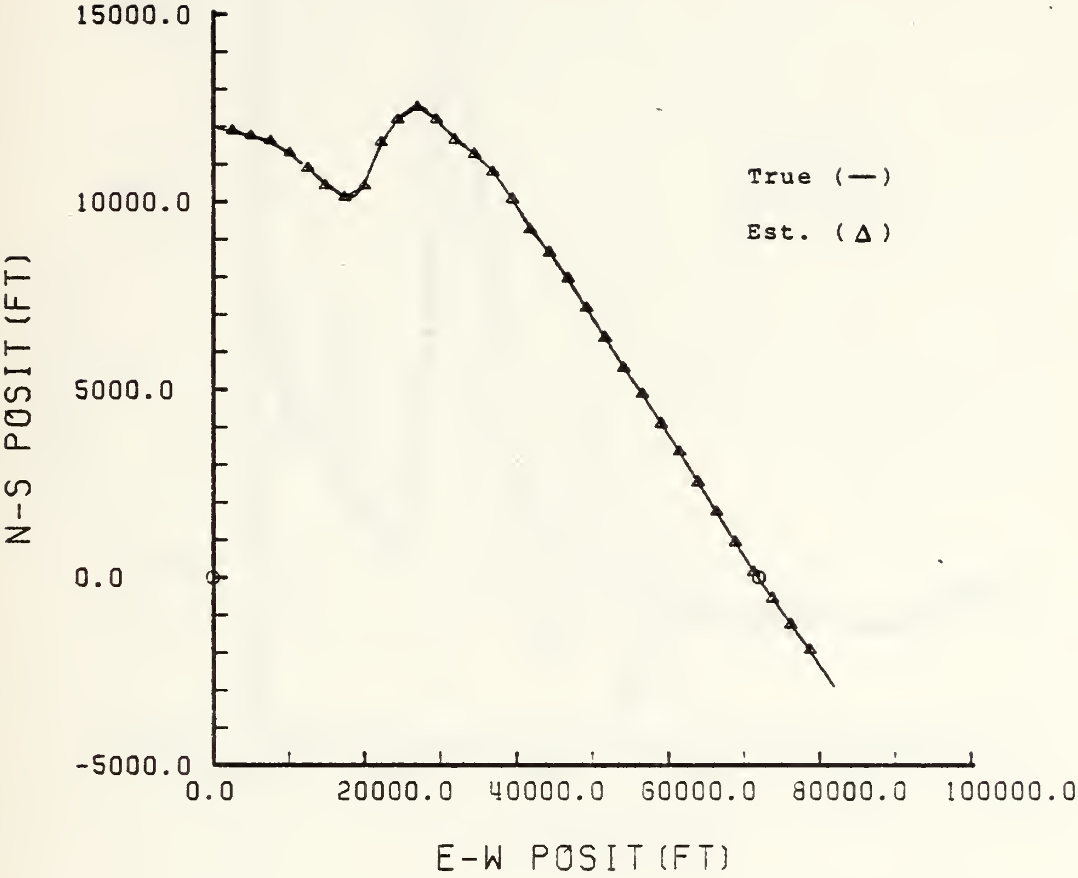


Fig. 4.19. KALMN2(K22) Horizontal Position Trajectory

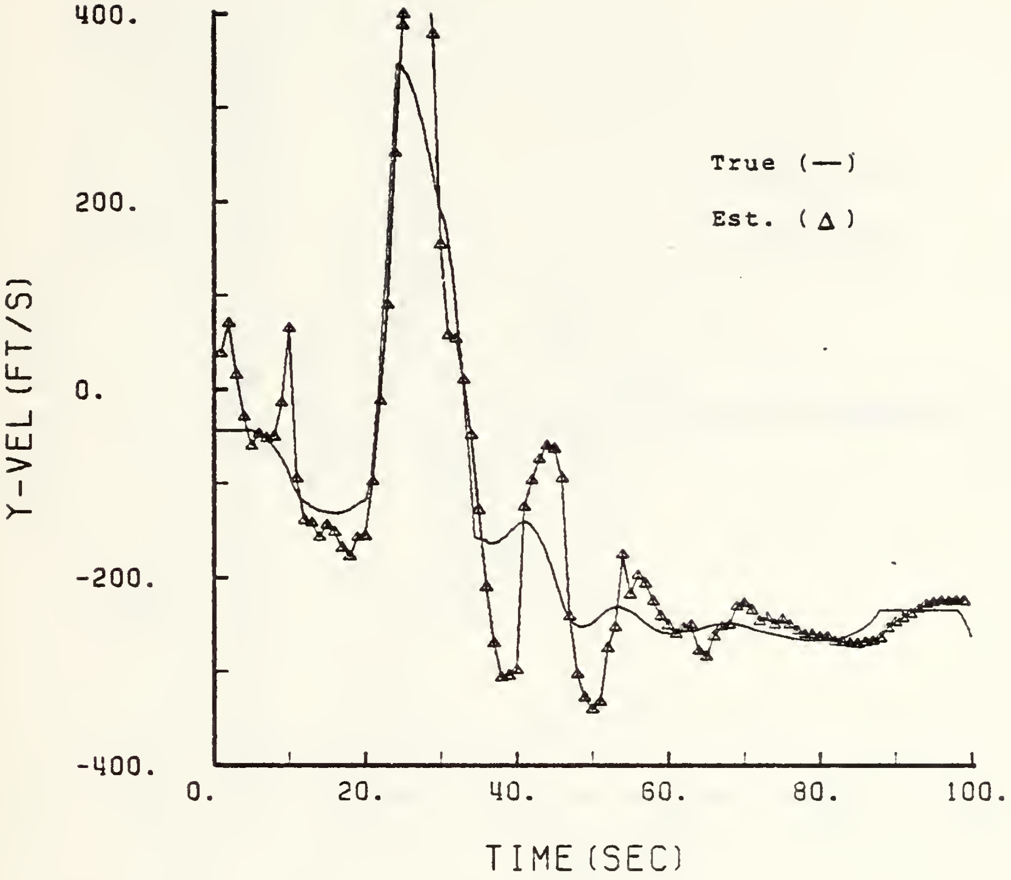


Fig. 4.20. KALMN2(K22) True and Estimated Y-Velocity

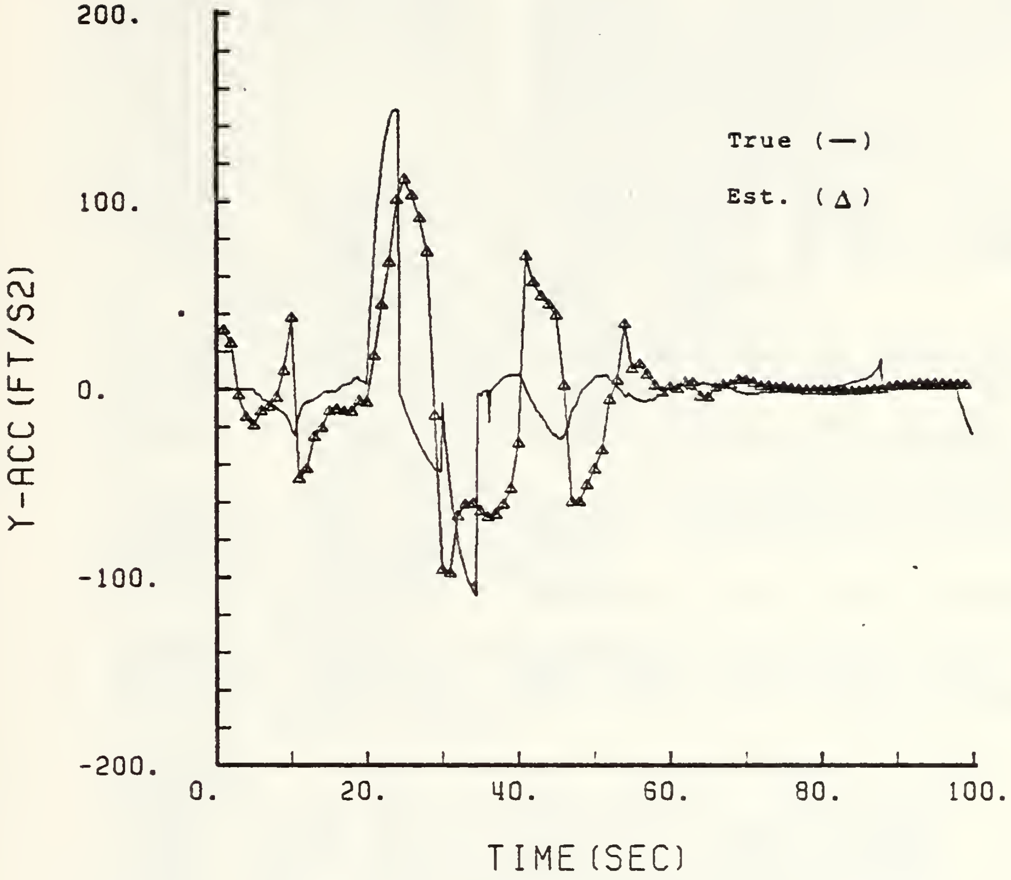


Fig. 4.21. KALMN (K22) True and Estimated Y-Acceleration

```
C********GROUND DIRECTED BOMBING SYSTEM SIMULATION**********
C
C
C
C
C
C
C
C
C     IMPORTANT VARIABLES DEFINED:
C        TACRRX(STATE,TIME)|  TRUE A/C STATE(1=POSIT,2=VEL,
C        TACRRY(STATE,TIME)|- 3=ACCEL) @ TIME(N) IN RADAR
C        TACRRZ(STATE,TIME)|  CARTESIAN COORD REF FRAME.
C        TACACX( "   ,  " )|  TRUE A/C STATE(",",") AT
C        TACACY( "   ,  " )|- TIME(N) IN A/C CARTESIAN COORD
C        TACACZ( "   ,  " )|  REFERENCE FRAME.
C        TACREX( "   ,  " )|  EST. AIRCRAFT STATE(",",") AT
C        TACREY( "   ,  " )|- TIME(N) IN A/C CARTESIAN COORD
C        TACREZ( "   ,  " )|  REFERENCE FRAME.
C        TACRMX - MEASURED A/C POSITION(X) IN RDR. FRAME
C        TACRMY - MEASURED A/C POSITION(Y) IN RDR. FRAME
C        TACRMZ - MEASURED A/C POSITION(Z) IN RDR. FRAME
C        RDSFQ - RADAR DATA SAMPLE FREQUENCY.
C        DT - RADAR SAMPLE INTERVAL TIME.
C        N - N-TH ITERATION AT SAMPLE RATE DT.
C        TACHDG - TRUE AIRCRAFT(AC) HEADING IN RADAR FRAME.
C        TACVEL - TRUE AIRCRAFT VELOCITY IN RADAR FRAME.
C        TTG - TIME TO GO UNTIL MOT TGT.
C        HDGERR - DIFFERENCE BETWEEN DSRDGT AND TACHDG.
C        ELPSTM - ELAPSE TIME IN SECONDS SINCE BEGINNING OF
C                 SIMULATION.
C        SRTMVR - START MANEUVERS;A/C MANEUVERING CAN BEGIN.
C        STPMVR - STOP MANEUVERS;A/C MANEUVERING MUST STOP.
C        MNVRPT - MANEUVERING PILOT.
C        CNTLPT - CONTROLED PILOT.
C        MODEL - FREE INERTIA MODEL OF A/C WITHOUT WIND
C                EFFECTS.
C        BNKANG - AIRCRAFT BANKANGLE.
C        NTIMES - NUMBER OF TIMES THRU SIMULATION LOOP.
C        GTRNG - RANGE(RNG) FROM A/C TO TGT ALONG DESIRED
C                GROUND TRACK.
C        TGTRRX|
C        TGTRRY|  -  TARGET COORD. IN RDR FRAME
C        TGTRRZ|
C        TWDRR - TRUE WIND DIRECTION IN RADAR FRAME
C        TWVRR - TRUE WIND VELOCITY IN RADAR FRAME
C        TWDRR - EST. WIND DIRECTION IN RADAR FRAME
C        TWVRR - EST. WIND VELOCITY IN RADAR FRAME
C        TWVRRX|  - TRUE WIND X-Y VELOCITY COMPONENTS
C        TWVRRY|
C        EWVRRX|  - EST. WIND X-Y VELOCITY COMPONENTS
C        EWVRRY|
C        X2SIGM|
C        Y2SIGM|  - PROCESS NOISE VARIANCE
C        Z2SIGM|
C        PINITL - ERROR COVARIANCE DIAGONAL INITIAL VALUE
C        GX|
C        GY|  - FILTER X,Y,Z GAIN VALUES AT TIME N
C        GZ|
C        XRES|
C        YRES|  - FILTER X,Y,Z RESIDUE VALUES AT TIME N
C        ZRES|
C        RADRES - RADIAL RESIDU AT TIME N
C
C
C
      IMPLICIT REAL(A-H,O-Z),INTEGER(I-N)
C
```

```
      DOUBLE PRECISION DSEED
      COMMON/DOUBLE/DSEED
      COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),TACRRZ(
     +3,1000)
      COMMON/TMACST/NT,TACACX(3,1000),TACACY(3,1000),TACACZ(
     +3,1000)
      COMMON/TGTWND/TGTRRX,TGTRRY,TGTRRZ,TWDRR,TWVRR,EWDRR,
     +EWVRR
      COMMON/PILOT/TB,SRTMVR,STPMVR,DT
      COMMON/NOISE/TACRMX(1000),TACRMY(1000),TACRMZ(1000)
      COMMON/FILTER/TACREX(3,1000),TACREY(3,1000),TACREZ(3,
     +1000)
      COMMON/PROCES/X2SIGM,Y2SIGM,Z2SIGM,EWVRRX,EWVRRY.GX(
     +1000),GY(1000),GZ(1000),RADRES(1000),XRES(1000)
     +,YRES(1000),ZRES(1000),W11(1000),W22(1000),W33(1000),
     +P(1000,9),PINITL
      COMMON/PARAMT/G,G2,GK1,GK2
      DIMENSION X(1000),Y(1000),TGTX(1),TGTY(1),TRUACC(1000)
     +,TIME(1000),X1(1000),Y1(1000),ESTACC(1000)
C
C
C
C
C     INITIALIZE PARAMETERS;
         BNKANG=0.0
         DEADZN=0.0
         OLDHDE=0.0
         K=0
         K1=0
         K2=1
         TB=2.0
         PINITL=1000.
         SRTMVR=20.0
         STPMVR=50.0
         IRDSFQ=8
         TTG=999.
         NTIMES=800
         SECOND=0.0
         HDGERR=0.0
         G=32.174049
         G2=16.087007
         GK1=3.5
         GK2=3.5
         PI=3.141592654
         OLDHDE=0.0
         DSEED= 1.0D0
         RSIGMA=100.
         ASIGMA=.001
         ESIGMA=.001
         KFILTR=0
         KADAPT=0
         X2SIGM=1.
         Y2SIGM=1.
         Z2SIGM=1.
C
C     READ INITIAL AIRCRAFT STATES IN AC REF. FRAME.
C
C
      WRITE(6,6)
      READ(5,1) TACACX(1,1),TACACY(1,1),TACACZ(1,1)
      READ(5,1) TACACX(2,1),TACACY(2,1),TACACZ(2,1)
      READ(5,1) TACACX(3,1),TACACY(3,1),TACACZ(3,1)
C
C
C     OBTAIN X & Y WIND COMPONENTS IN RADAR FRAME & ADD TO
C     CORRESPONDING A/C INITIAL VELOCITY STATES TO OBTAIN
C     INITIAL AIRCRAFT STATES IN RADAR REFERENCE FRAME.
C
```

```
      READ(5,2) TGTRRX,TGTRRY,TGTRRZ,TWDRR,TWVRR,EWDRR,EWVRR
      TWVRRX=TWVRR*SIN(TWDRR*PI/180.0)*1.687805556
      TWVRRY=TWVRR*COS(TWDRR*PI/180.0)*1.687805556
      EWVRRX=EWVRR*SIN(EWDRR*PI/180.0)*1.687805556
      EWVRRY=EWVRR*COS(EWDRR*PI/180.0)*1.687805556
      DO 41 I=1,3
         IF (I .NE. 2) GO TO 3
            TACRRX(2,1)=TACACX(2,1) + TWVRRX
            TACRRY(2,1)=TACACY(2,1) + TWVRRY
            TACRRZ(2,1)=TACACZ(2,1)
         GO TO 41
 3       CONTINUE
         TACRRX(I,1)=TACACX(I,1)
         TACRRY(I,1)=TACACY(I,1)
         TACRRZ(I,1)=TACACZ(I,1)
 41   CONTINUE
C
      WRITE(6,4)
      WRITE(6,6)
      WRITE(6,8)  TACRRX(1,1),TACRRY(1,1),TACRRZ(1,1)
      WRITE(6,10) TACRRX(2,1),TACRRY(2,1),TACRRZ(2,1)
      WRITE(6,12) TWDRR,TWVRR
      WRITE(6,14) EWDRR,EWVRR
      WRITE(6,16) RSIGMA,ASIGMA,ESIGMA
      WRITE(6,18) X2SIGM,Y2SIGM,Z2SIGM
      WRITE(6,20) PINITL
      WRITE(6,21) KFILTR
      WRITE(6,22) KADAPT
      WRITE(6,24) SRTMVR
      WRITE(6,26) STPMVR
      WRITE(6,50)
      WRITE(6,53)
      WRITE(6,60)
      WRITE(6,63)
      WRITE(6,65)
C
C
C     DETERMINE RADAR SAMPLE INTERVAL(DT).
      DT=1.0/FLOAT(IRDSFQ)
C
C     PERFORM SIMULATION LOOP N-TIMES.
      DO 900 N = 1,NTIMES
         NT=N
         ELPSTM = (N-1) * DT
         TIME(N)=ELPSTM
C        NOW CALL PLRNOI TO TRANSFORM RADAR CARTESIAN COORDS
C        IN POLAR, ADD WHITE NOISE, AND RETRANSFORM INTO RDR
C        CARTESIAN COORDS IN PREPARATION FOR FILTERING.
C
C
         CALL PLRNOI(RSIGMA,ASIGMA,ESIGMA,RANGE,AZMITH,
     +ELEVTN)
C
C
C
         IF (KFILTR .NE. 0) GO TO 45
         CALL ALFBTA(DT,HDGERR,TTG)
         GO TO 59
 45      CONTINUE
         IF (KFILTR .NE. 1) GO TO 46
         CALL KALMN1(KADAPT,DT,RSIGMA,ASIGMA,ESIGMA,RANGE,
     +AZMITH,ELEVTN,TTG)
         GO TO 59
 46      CONTINUE
         CALL KALMN2(KADAPT,DT,RSIGMA,ASIGMA,ESIGMA,RANGE,
     +AZMITH,ELEVTN,TTG)
 59      CONTINUE
```

```
C
C
C        NOW CALL TACAN SUBROUTINE TO OBTAIN HEADING TO FLY
C        TO TGT.
         CALL TACAN(TACHDG,DSRDGT,TACVEL,TTG,GTRNG,HDGERR)
C
         IF (N .GT. 800) GO TO 580
            IF (MOD(N-1,8) .NE. 0) GO TO 550
               WRITE(6,75) ELPSTM,TACRRX(1,N),TACRRY(1,N),
     +TACRRZ(1,N),TACRRX(2,N),TACRRY(2,N),TACRRZ(2,N),
     +TACRRX(3,N),TACRRY(3,N),TACRRZ(3,N)
               WRITE(6,78) TACREX(1,N),TACREY(1,N),TACREZ(
     +1,N),TACREX(2,N),TACREY(2,N),TACREZ(2,N),
     +TACREX(3,N),TACREY(3,N),TACREZ(3,N)
               WRITE(6,78) TTG,TACRMX(N),TACRMY(N),TACRMZ(N)
     +,HDGERR,DSRDGT,TACHDG,TACVEL,GTRNG
               WRITE(6,78) GX(N),GY(N),GZ(N),XRES(N),YRES(N)
     +,ZRES(N),W11(N),W22(N),W33(N)
               WRITE(6,78) P(N,1),P(N,2),P(N,3),P(N,4),
     +P(N,5),P(N,6),P(N,7),P(N,8),P(N,9)
 550        CONTINUE
 580     CONTINUE
C
C        NO CONTROL INPUT UNTIL FILTER SETTLES.
         IF (ELPSTM .LT. 5.) GO TO 800
C
C        DETERMINE CONTROL INPUT TO AC.
C
C        NO CONTROL INPUT IF TTG IS LESS THAN 3 SECOND OR IF
C        CLOSEST POINT OF APPROACH(CPA) HAS BEEN REACHED.
C
         CALL TTGCPA(N,K,TTG,GTRNG,IFLAG)
C
C
         IF (((IFLAG) .EQ. 1) .AND. (ABS(HDGERR) .GT. DEADZN
     +)) GO TO 600
            TACACX(3,N)=0.0
            TACACY(3,N)=0.0
            TACACZ(3,N)=0.0
         GO TO 800
C
C        CHOOSE MNVRPT OR CNTLPT RESPONSE BASED ON
C        MAGNITUDE OF HDGERR,AND SRTMVR AND STPMVR CRITERIA.
C
 600     CONTINUE
         IF ((ABS(HDGERR)) .GT. .523598776) GO TO 700
            IF ((ELPSTM .LT. SRTMVR).OR. (TTG .LT. STPMVR))
     +GO TO 700
               CALL MNVRPT(K1)
            GO TO 800
C
C        CALL CNTLPT SUBROUTINE IF MANEUVER NOT PERMITTED.
C
C
 700     CONTINUE
         CALL CNTLPT(K2,HDGERR)
C
C        PROVIDE MODEL SUBROUTINE AC STATES AT TIME N.
C
 800     CALL XMODEL
C
C        MODEL RETURNS STATES IN AC FRAME FOR TIME N+1.
C
C        ADD WIND EFFECTS OVER DT TO OBTAIN AC STATES IN
C        RADAR FRAME.
         CALL WNDEFF(TWVRRX,TWVRRY,DT)
C
```

```
900   CONTINUE
      STOP
1     FORMAT(3F10.0)
2     FORMAT(7F10.0)
4     FORMAT(58X,'GCBS SIMULATION')
6     FORMAT(////2X,'INITIAL CONDITIONS:')
8     FORMAT(//5X,'A/C INITIAL POSITION IN RADAR REFERENCE
     +SYSTEM(FT) =',3(10X,F10.3)/)
10    FORMAT(5X,'A/C INITIAL VELOCITY IN RADAR REFERENCE
     +SYSTEM(FT) =',3(10X,F10.3)//)
12    FORMAT(5X,'TRUE WIND =',10X,F10.3,'/',1X,F9.3/)
14    FORMAT(5X,'EST. WIND =',10X,F10.3,'/',1X,F9.3//)
16    FORMAT(5X,'MEASUREMENT NOISE VARIANCES(R(FT),A(RAD),
     +E(RAD)) =',3(10X,F10.3)/)
18    FORMAT(5X,'PROCESS NOISE VARIANCES(X,Y,Z) =',16X,
     +3(10X,F10.3)/)
20    FORMAT(5X,'ERROR COVARIANCE DIAGONAL =',10X,F10.3//)
21    FORMAT(5X,'FILTER DESIGNATION =',4X,I1/)
22    FORMAT(5X,'FILTER ADAPTATION =',5X,I1//)
24    FORMAT(5X,'START MANEUVER =',10X,F10.3/)
26    FORMAT(5X,'STOP MANEUVER  =',10X,F10.3//)
50    FORMAT('0',6X,'SECOND',6X,'TACRRX1',6X,'TACRRY1',6X,
     +'TACRRZ1',6X,'TACRRX2',6X,'TACRRY2',6X,'TACRRZ2',6X,
     +'TACRRX3',6X,'TACRRY3',6X,'TACRRZ3')
53    FORMAT('0',18X,'TACREX1',6X,'TACREY1',6X,'TACREZ1',6X
     +,'TACREX2',6X,'TACREY2',6X,'TACREZ2',6X,
     +'TACREX3',6X,'TACREY3',6X,'TACREZ3')
60    FORMAT('0',22X,'TTG',7X,'TACRMX',7X,'TACRMY',7X,
     +'TACRMZ',7X,'HDGERR',7X,'DSRDGT',7X,
     +'TACHDG',7X,'TACVEL',8X,'GTRNG')
63    FORMAT('0',19X,'X-GAIN',7X,'Y-GAIN',7X,'Z-GAIN',5X,
     +'X-RESIDU',5X,'Y-RESIDU',5X,'Z-RESIDU',7X,'W(1,1)'
     +,7X,'W(2,2)', 7X,'W(3,3)')
65    FORMAT('0',20X,'PKK11',8X,'PKK22',8X,'PKK33', 8X,
     +'PKK44',8X,'PKK55',8X,'PKK66',8X,'PKK77',8X,'PKK88'
     +,8X,'PKK99'///)
75    FORMAT(///10F13.4)
78    FORMAT(13X,9F13.4)
      END
      SUBROUTINE TACAN(TACHDG,DSRDGT,TACVEL,TTG,GTRNG,
     +HDGERR)
          IMPLICIT REAL(A-H,O-Z),INTEGER(I-N)
C
          DOUBLE PRECISION DSEED
          COMMON/DOUBLE/DSEED
          COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
          COMMON/TGTWND/TGTRRX,TGTRRY,TGTRRZ,TWDRR,TWVRR,
     +EWDRR,EWVRR
          COMMON/PILOT/TB,SRTMVR,STPMVR,DT
          COMMON/FILTER/TACREX(3,1000),TACREY(3,1000),
     +TACREZ(3,1000)
          COMMON/PARAMT/G,G2,GK1,GK2
C
          PI=3.141592654
          EWD=EWDRR*PI/180.0
          EWV=EWVRR*1.68780556
          TACVEL=SQRT(TACREX(2,N)*TACREX(2,N)+TACREY(2,N)*
     +TACREY(2,N))
          TACHDG=TRUHDG(TACREX(2,N),TACREY(2,N))
          TGTDIR=TRUHDG((TGTRRX-TACREX(1,N)),(TGTRRY-
     +TACREY(1,N)))
          DSRDGT=TGTDIR
          HDGERR=DSRDGT-TACHDG
          GTRNG=SQRT((TACREX(1,N)-TGTRRX)**2+(TACREY(1,N)-
     +TGTRRY)**2)
          TTG=GTRNG/TACVEL
```

```
         RETURN
      END
C
C
C
      SUBROUTINE TTGCPA(N,K,TTG,RNG,IFLAG)
         IF ((TTG .GT. 3.0) .OR. (K .EQ. 0)) GO TO 600
            GO TO 700
 600     CONTINUE
         IF (N .LT. 100)  OLDRNG=RNG
         DIFFNC=OLDRNG-RNG
         IF (TTG .GT. 10.) GO TO 800
         IF (DIFFNC) 700,800,800
 700        IFLAG=-1
            K=1
            GO TO 900
 800        IFLAG=1
 900     CONTINUE
         OLDRNG=RNG
         RETURN
      END
C
C
C
      SUBROUTINE MNVRPT(K1)
         DOUBLE PRECISION DSEED
         COMMON/DOUBLE/DSEED
         COMMON/TMACST/N,TACACX(3,1000),TACACY(3,1000),
     +TACACZ(3,1000)
         COMMON/PILOT/TB,SRTMVR,STPMVR,DT
         COMMON/PARAMT/G,G2,GK1,GK2
         IF ((N-K1) .EQ. 1) GO TO 100
 90         CMDACC=0.0
            N1=0
            PA=GGUBFS(DSEED)
            AMXACC=14.0*(PA-.5)
            ACCDUR=20.0*EXP(.25*ABS(AMXACC))
 100     CONTINUE
         ACCDUR=ACCDUR-DT
         IF (ACCDUR .LE. 0.0) GO TO 90
         CMDACC=AMXACC*(1.-EXP(-DT/TB))+CMDACC*EXP(-DT/TB)
         AACHDG=TRUHDG(TACACX(2,N),TACACY(2,N))
         TACACX(3,N)=CMDACC*COS(AACHDG)*G
         TACACY(3,N)=-CMDACC*SIN(AACHDG)*G
         TACACZ(3,1000)=0.0
         K1=N
         RETURN
      END
C
C
C
      SUBROUTINE CNTLPT(K2,HDGERR)
C        GENERATE CONTROLED PILOT BANKANGLE.
C        PHD1 IS COMPONENT BASED ON ANGLE ERROR HDE.
C        PHD2 IS COMPONENT BASED ON ANGLE ERROR RATE HDEDOT.
C        PHD IS DESIRED PILOT GENERATED BANK ANGLE.
         DOUBLE PRECISION DSEED
         COMMON/DOUBLE/DSEED
         COMMON/TMACST/N,TACACX(3,1000),TACACY(3,1000),
     +TACACZ(3,1000)
         COMMON/PARAMT/G,G2,GK1,GK2
         COMMON/PILOT/TB,SRTMVR,STPMVR,DT
         IF ((N-K2) .EQ. 1) GO TO 200
            OLDHDE=0.0
            BNKANG=0.0
            N2=0
 200     CONTINUE
```

```
          N2=N2+1
          HDEDOT=(HDGERR-OLDHDE)/DT
          PHD1=GK1*HDGERR
          PHD2=GK2*HDEDOT
          CMDANG=PHD1+PHD2
          IF (ABS(HDGERR) .GT. .1745) GO TO 202
             IF (CMDANG .GE. .5236)  CMDANG=.5236
             IF (CMDANG .LE. -.5236)  CMDANG=-.5236
          GO TO 204
 202      CONTINUE
             IF (CMDANG .GE. 1.047)  CMDANG=1.047
             IF (CMDANG .LE. -1.047)  CMDANG=-1.047
 204      CONTINUE
          BNKANG=CMDANG*(1.-EXP(-DT/TB))+BNKANG*EXP(-DT/TB)
          AACVEL=SQRT(TACACX(2,N)*TACACX(2,N)+TACACY(2,N)*
     +TACACY(2,N))
          AACHDG=TRUHDG(TACACX(2,N),TACACY(2,N))
          HDGRAT=TAN(BNKANG)*G/AACVEL
          TACACX(3,N)=AACVEL*HDGRAT*COS(AACHDG)
          TACACY(3,N)=-AACVEL*HDGRAT*SIN(AACHDG)
          TACACZ(3,N)=0.0
          OLDHDE=HDGERR
          K2=N
          RETURN
       END
C
C
C
       SUBROUTINE XMODEL
          DOUBLE PRECISION DSEED
          COMMON/DOUBLE/DSEED
          COMMON/TMACST/N,TACACX(3,1000),TACACY(3,1000),
     +TACACZ(3,1000)
          COMMON/PILOT/TB,SRTMVR,STPMVR,DT
          COMMON/PROCES/X2SIGM,Y2SIGM,Z2SIGM
          COMMON/PARAMT/G,G2,GK1,GK2
          DIMENSION PHI(6,6),DEL(6,3),X(6,1),U(3,1),F(6,1),
     +XN1(6,1),X1(6,1)
          T2=(DT*DT)/2
          IF (N .NE. 1) GO TO 160
C            COMPUTE STATE TRANSITION MATRIX(PHI).
C            INITIALIZE MATRIX @ 0.0 , 1.0 ON DIAGONAL.
             DO 130 I=1,6
                DO 120 J=1,6
                   PHI(I,J)=0.0
                   IF (I .EQ. J)  PHI(I,J)=1.0
 120            CONTINUE
 130         CONTINUE
             PHI(1,2)=DT
             PHI(3,4)=DT
             PHI(5,6)=DT
C            COMPUTE DEL MATRIX(DEL)
C            INITIALIZE @ 0.0.
             DO 150 I=1,6
                DO 140 J=1,3
                   DEL(I,J)=0.0
 140            CONTINUE
 150         CONTINUE
             DEL(1,1)=T2
             DEL(3,2)=T2
             DEL(5,3)=T2
             DEL(2,1)=DT
             DEL(4,2)=DT
             DEL(6,3)=DT
C
C
C            SET UP 'X' MATRIX.
```

```
 160        X(1,1)=TACACX(1,N)
            X(2,1)=TACACX(2,N)
            X(3,1)=TACACY(1,N)
            X(4,1)=TACACY(2,N)
            X(5,1)=TACACZ(1,N)
            X(6,1)=TACACZ(2,N)
C
C           CALL PRODUCT SUBROUTINE TO MULTIPLY PHI & X.
            CALL PRODCT(PHI,X,6,1,6,X1)
C
C           SET UP 'U' MATRIX.
C
            U(1,1)=TACACX(3,N)
            U(2,1)=TACACY(3,N)
            U(3,1)=TACACZ(3,N)
C
C           CALL PRODUCT SUBROUTINE TO MULTIPLY DEL & U.
            CALL PRODCT(DEL,U,3,1,6,F)
C
C           CALL ADD SUBROUTINE TO ADD X & F TO OBTAIN
C           STATES AT TIME (N+1).
C
            CALL ADD(X1,F,1,6,XN1)
C
C
            IP=N+1
            TACACX(1,IP)=XN1(1,1)
            TACACX(2,IP)=XN1(2,1)
            TACACX(3,IP)=TACACX(3,N)
            TACACY(1,IP)=XN1(3,1)
            TACACY(2,IP)=XN1(4,1)
            TACACY(3,IP)=TACACY(3,N)
            TACACZ(1,IP)=XN1(5,1)
            TACACZ(2,IP)=XN1(6,1)
            TACACZ(3,IP)=TACACZ(3,N)
C
            RETURN
      END
C
C
C
C THIS SUBROUTINE COMPUTES THE MATRIX PRODUCT A*B AND
C STORES RESULT IN C.
      SUBROUTINE PRODCT(A,B,N,M,L,C)
      DIMENSION A(L,N),B(N,M),C(L,M)
C
C        INITIALIZE 'C' MATRIX@ 0.0.
         DO 180 I=1,L
            DO 170 J=1,M
               C(I,J)=0.0
 170        CONTINUE
 180     CONTINUE
C
C        MULTIPLY.
C
         DO 210 I=1,L
            DO 200 J=1,N
               DO 190 K=1,M
                  C(I,K)=C(I,K)+A(I,J)*B(J,K)
 190           CONTINUE
 200        CONTINUE
 210     CONTINUE
         RETURN
      END
C
C
C
```

```
      SUBROUTINE ADD(A,B,L,M,C)
         DIMENSION A(M,L),B(M,L),C(M,L)
C        INITIALIZE 'C' @ 0.0.
         DO 215 I=1,M
            DO 213 J=1,L
               C(I,J)=0.0
 213        CONTINUE
 215     CONTINUE
C
C        NOW ADD 'A' & 'B'.
C
         DO 230 I=1,M
            DO 220 J=1,L
               C(I,J)=A(I,J)+B(I,J)
 220        CONTINUE
 230     CONTINUE
         RETURN
      END
C
C
C
      SUBROUTINE WNDEFF(TWVRRX,TWVRRY,DT)
         COMMON/TMACST/NT,TACACX(3,1000),TACACY(3,1000),
     +TACACZ(3,1000)
         COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
         IP=N+1
         TACRRX(1,IP)=TACACX(1,IP)+DT*TWVRRX*N
         TACRRX(2,IP)=TACACX(2,IP)+TWVRRX
         TACRRX(3,IP)=TACACX(3,IP)
         TACRRY(1,IP)=TACACY(1,IP)+DT*TWVRRY*N
         TACRRY(2,IP)=TACACY(2,IP)+TWVRRY
         TACRRY(3,IP)=TACACY(3,IP)
         TACRRZ(1,IP)=TACACZ(1,IP)
         TACRRZ(2,IP)=TACACZ(2,IP)
         TACRRZ(3,IP)=TACACZ(3,IP)
         RETURN
      END
C
C
C
      FUNCTION TRUHDG(XVEL,YVEL)
         PI=3.141592654
         IF (XVEL) 31,41,51
 31      IF (YVEL) 71,111,71
 41      IF (YVEL) 101,91,121
 51      IF (YVEL) 61,91,61
 61         TRUHDG=PI/2.0 - ATAN(YVEL/XVEL)
            GO TO 131
 71         TRUHDG=3.0*PI/2.0 - ATAN(YVEL/XVEL)
            GO TO 131
 91         TRUHDG=PI/2.0
            GO TO 131
 101        TRUHDG=PI
            GO TO 131
 111        TRUHDG=3.0*PI/2.0
            GO TO 131
 121        TRUHDG=0.0
 131     RETURN
      END
C
C
C
C
C     THIS SUBROUTINE TRANSFORMS CARTESIAN COORDS INTO
C     POLAR IN THE RDR REF. FRAME THEN ADDS WHITE NOISE OF
C     0 MEAN AND VARIANCE, RSIGMA,ASIGMA, AND ESIGMA TO THE
C     RANGE, AZIMITH, AND ELEVATION RESPECTIVELY. THESE
```

```
C     CONSTITUTE THE MEASURED POSIT OF THE A/C. THESE NOISY
C     POLAR MEASUREMENTS ARE THEN TRANSFORMED BACK INTO THE
C     RADAR CARTESIAN SYSTEM IN PREPARATION FOR FILTERING.
C
C
      SUBROUTINE PLRNOI(RSIGMA,ASIGMA,ESIGMA,RNGPNS,AZIPNS,
     +ELVPNS)
         DOUBLE PRECISION DSEED
         COMMON/DOUBLE/DSEED
         COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
         COMMON/NOISE/TACRMX(1000),TACRMY(1000),TACRMZ(1000)
         COMMON/PARAMT/G,G2,GK1,GK2
C
C
C
         SLNTRG=SQRT(TACRRX(1,N)**2+TACRRY(1,N)**2+
     +TACRRZ(1,N)**2)
         AZMITH=TRUHDG(TACRRX(1,N),TACRRY(1,N))
         GNDRNG=SQRT(TACRRX(1,N)**2+TACRRY(1,N)**2)
         ELVATN=ATAN(TACRRZ(1,N)/GNDRNG)
         R=GGNQF(DSEED)
         RNGPNS=SLNTRG+R*RSIGMA
         A=GGNQF(DSEED)
         AZIPNS=AZMITH+A*ASIGMA
         E=GGNQF(DSEED)
         ELVPNS=ELVATN+E*ESIGMA
         TACRMX(N)=RNGPNS*COS(ELVPNS)*SIN(AZIPNS)
         TACRMY(N)=RNGPNS*COS(ELVPNS)*COS(AZIPNS)
         TACRMZ(N)=RNGPNS*SIN(ELVPNS)
         RETURN
      END
C
C
C
C
      SUBROUTINE ALFBTA(DT,HDGERR,TTG)
         COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
         COMMON/NOISE/TACRMX(1000),TACRMY(1000),TACRMZ(1000)
         COMMON/FILTER/TACREX(3,1000),TACREY(3,1000),
     +TACREZ(3,1000)
         COMMON/PROCES/X2SIGM,Y2SIGM,Z2SIGM,EWVRRX,EWVRRY,
     +GX(1000),GY(1000),GZ(1000),RADRES(1000),XRES(1000),
     +YRES(1000),ZRES(1000),W11(100),W22(1000),W33(1000),
     +P(1000,9),PINITL
C
         DT2=DT*DT
C
         IF (N .NE. 1) GO TO 10
C
C     INITIALIZE FILTER.
         FPXKK=TACRMX(N)
         FPYKK=TACRMY(N)
         FPZKK=TACRMZ(N)
         FVXKK=TACRRX(2,N)+EWVRRX
         FVYKK=TACRRY(2,N)+EWVRRY
         FVZKK=TACRRZ(2,N)
         FPXKK1=FPXKK+DT*FVXKK
         FPYKK1=FPYKK+DT*FVYKK
         FPZKK1=FPZKK+DT*FVZKK
         STRGER=0.0
         STRGRT=0.0
         MVRFLG=0
         MVRCNT=0.0
C
C
```

```
      GO TO 999
   10 CONTINUE
C
C
C     SELECT PARAMETERS FOR FILTERING BASED ON
C     MANEUVERING AND NOISE CRITERIA.
C
C     HIGH OR LOW NOISE ENVIRONMENT?
C
      RUFRNG=SQRT(TACRMX(N)**2+TACRMY(N)**2+TACRMZ(N)**2)
      IF (RUFRNG .GT. 1500.) GO TO 20
         NOSFLG=0
         GO TO 30
   20 CONTINUE
      NOSFLG=1
   30 CONTINUE
C
C     MANEUVERING?
C     IF TTG .LT. 30 SEC ASSUME NO MANEUVERING AND GO ON.
C
      IF (TTG .LT. 30.) GO TO 40
         CALL STRG(DT,NOSFLG,HDGERR,STRGER,STRGRT)
         CALL TMNVR(STRGRT,MVRCNT)
         IF ((DT*FLOAT(MVRCNT)) .GT. 5.0) MVRFLG=0
         IF ((DT*FLOAT(MVRCNT)) .LT. -5.0) MVRFLG=0
         IF (ABS(HDGERR) .GT. .052359878) MVRFLG=1
         GO TO 50
   40 CONTINUE
      MVRFLG=0
   50 CONTINUE
C
C     FIND ALFA/BETA PARAMETERS .
      CALL CRSTRK(NOSFLG,MVRFLG,TTG,CALFA,CBETA)
      GX(N)=CALFA
C
C
C
      GY(N)=CALFA
      GZ(N)=CALFA
C
C     FILTER X-COORD. DATA.
C     CURRENT STATES.
C
C
      XRESDU=TACRMX(N)-FPXKK1
      FPXKK=FPXKK1+CALFA*XRESDU
      FVXKK=FVXKK+CBETA*XRESDU/DT
C
C     PREDICTED STATES.
C
      FPXKK1=FPXKK+FVXKK*DT
C
C     FILTER Y-COORD. DATA.
C     CURRENT STATES.
C
C
      YRESDU=TACRMY(N)-FPYKK1
      FPYKK=FPYKK1+CALFA*YRESDU
      FVYKK=FVYKK+CBETA*YRESDU/DT
C
C     PREDICTED STATES.
C
      FPYKK1=FPYKK+FVYKK*DT
C
C     FILTER Z-COORD. DATA.
C     CURRENT STATES.
C
```

```
C
      ZRESDU=TACRMZ(N)-FPZKK1
      FPZKK=FPZKK1+CALFA*ZRESDU
      FVZKK=FVZKK+CBETA*ZRESDU/DT
C
C     PREDICTED STATES.
C
      FPZKK1=FPZKK+FVZKK*DT
C
      XRES(N)=XRESDU
      YRES(N)=YRESDU
      ZRES(N)=ZRESDU
      RADRES(N)=SQRT(XRESDU**2+YRESDU**2+ZRESDU**2)
 999  CONTINUE
      TACREX(1,N)=FPXKK
      TACREY(1,N)=FPYKK
      TACREZ(1,N)=FPZKK
      TACREX(2,N)=FVXKK
      TACREY(2,N)=FVYKK
      TACREZ(2,N)=FVZKK
      RETURN
      END
C
C
C
C
      SUBROUTINE STRG(DT,NOSFLG,HDGERR,STRGER,STRGRT)
         IF (NOSFLG .EQ. 1) GO TO 15
            ALFA=.0625
            BETA=.0078125
         GO TO 20
 15      CONTINUE
            ALFA=.03125
            BETA=.001953125
 20      CONTINUE
         STRGER=STRGER+ALFA*(HDGERR-STRGER)
         STRGRT=STRGRT+BETA*(HDGERR-STRGER)
         STRGER=STRGER+STRGRT*DT
         RETURN
      END
C
C
C
      SUBROUTINE TMNVR(STRGRT,MVRCNT)
         IF (ABS(STRGRT) .GT. .00872665) GO TO 10
            IF (MVRCNT.LT. 0) MVRCNT=0
            MVRCNT=MVRCNT+1
         GO TO 30
 10      IF (ABS(STRGRT) .LT. .0174533) GO TO 20
            IF (MVRCNT .GT. 0) MVRCNT=0
            MVRCNT=MVRCNT-1
         GO TO 30
 20      MNRCNT=0
 30      CONTINUE
         RETURN
      END
C
C
C
C
      SUBROUTINE CRSTRK(NOSFLG,MVRFLG,TTG,CALFA,CBETA)
         IF (NOSFLG .EQ. 0) GO TO 50
            IF (MVRFLG .EQ. 0) GO TO 10
C              LOW PRECISION (MANEUVERING)
               CALFA=.1875
               CBETA=.0078125
               GO TO 30
```

```
 10         CONTINUE
C           HIGH PRECISION (NON-MANUEVERING)
            IF (TTG .LT. 30.)GO TO 20
C              MANUEVERING STOPPED WITH .GT. 30 SEC TO GO.
               CALFA=.0390625
               CBETA=.0002441406
               GO TO 30
 20         CONTINUE
C              MANUEVERING STOPPED WITH .LT. 30. SEC TO GO.
               CALFA=.078125
               CBETA=.0009765625
 30         GO TO 99
C        LOW NOISE.
 50      CONTINUE
            IF (MVRFLG .EQ. 0) GO TO 60
C              LOW PRECISION (MANEUVERING)
               CALFA=.375
               CBETA=.03125
               GO TO 80
 60         CONTINUE
C              HIGH PRECISION (NON-MANEUVERING)
               IF (TTG .LT. 30.) GO TO 70
C                 MNVRNG STOPPED WITH 30 OR MORE SEC TTG.
                  CALFA=.078125
                  CBETA=.0009765625
                  GO TO 80
 70            CONTINUE
C                 MNVRNG STOPPED WITH LESS THAN 30 SEC TTG.
                  CALFA=.15625
                  CBETA=.00390625
 80            CONTINUE
 99         CONTINUE
         RETURN
      END
C
C
      SUBROUTINE KALMN1(KAD,DT,RSIGMA,ASIGMA,ESIGMA,RANGE,
     +AZMITH,ELEVTN,TTG)
C
C
C
      DOUBLE PRECISION DSEED
      COMMON/DOUBLE/DSEED
      COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
      COMMON/NOISE/TACRMX(1000),TACRMY(1000),TACRMZ(1000)
      COMMON/FILTER/TACREX(3,1000),TACREY(3,1000),
     +TACREZ(3,1000)
      COMMON/PROCES/X2SIGM,Y2SIGM,Z2SIGM,EWVRRX,EWVRRY,
     +GX(1000),GY(1000),GZ(1000),RADRES(1000),XRES(1000),
     +YRES(1000),ZRES(1000),W11(1000),W22(1000)
     +,W33(1000),P(1000,9),PINITL
C
      DIMENSION Q(6,6),PKK(6,6),R(6,6),W(6,6),G(6,6),
     +PHI(6,6),DEL(6,6),DI(6,6),NULL(6,6),H(6,6),XKK(6),
     +XKK1(6),FK1(6),EXTRA1(6,6),EXTRA2(6,6),PKK1(6,6),
     +WKAREA(18),EXTRA3(3,3),EXTRA4(3,3),Z(6)
C
C
C
      IF (N .NE. 1 ) GO TO 25
      IDGT=2
      DT2=DT*DT/2.0
C
C
      DO 20 I=1,6
         Z(I)=0.0
```

```
         DO 10 J=1,6
            Q(I,J)=0.0
            PKK(I,J)=0.0
               IF (I .EQ. J) PKK(I,J)=PINITL
            R(I,J)=0.0
            W(I,J)=0.0
            G(I,J)=0.0
            PHI(I,J)=0.0
            DEL(I,J)=0.0
            DI(I,J)=0.0
               IF (I .EQ. J) DI(I,J)=1.0
            NULL(I,J)=0.0
            H(I,J)=0.0
 10      CONTINUE
 20   CONTINUE
      W1=X2SIGM*X2SIGM
      W2=Y2SIGM*Y2SIGM
      W3=Z2SIGM*Z2SIGM
      WX=0.0
      WY=WX
      WZ=WY
      IXFLG=0
      IYFLG=0
      IZFLG=0
      DEL(1,1)=DT2
      DEL(2,1)=DT
      DEL(3,2)=DT2
      DEL(4,2)=DT
      DEL(5,3)=DT2
      DEL(6,3)=DT
      H(1,1)=1.0
      H(2,3)=1.0
      H(3,5)=1.0
      PHI(1,1)=1.0
      PHI(1,2)=DT
      PHI(2,2)=1.0
      PHI(3,3)=1.0
      PHI(3,4)=DT
      PHI(4.4)=1.0
      PHI(5,5)=1.0
      PHI(5,6)=DT
      PHI(6,6)=1.0
      RVAR=RSIGMA*RSIGMA
      AVAR=ASIGMA*ASIGMA
      EVAR=ESIGMA*ESIGMA
      TACREX(1,1)=TACRMX(1)
      XKK(1)=TACREX(1,1)
      TACREY(1,1)=TACRMY(1)
      XKK(3)=TACREY(1,1)
      TACREZ(1,1)=TACRMZ(1)
      XKK(5)=TACREZ(1,1)
      TACREX(2,1)=TACRRX(2,N)+EWVRRX
      XKK(2)=TACREX(2,1)
      TACREY(2,1)=TACRRY(2,N)+EWVRRY
      XKK(4)=TACREY(2,1)
      TACREZ(2,1)=TACRRZ(2,N)
      XKK(6)=TACREZ(2,1)
C
      IF (N .EQ. 1) GO TO 999
C
C
 25   CONTINUE
C
      Z(1)=TACRMX(N)
      Z(2)=TACRMY(N)
      Z(3)=TACRMZ(N)
C
```

```
C
C
      R2=RANGE*RANGE
      CA=COS(AZMITH)
      SA=SIN(AZMITH)
      CE=COS(ELEVTN)
      SE=SIN(ELEVTN)
      CA2=CA*CA
      SA2=SA*SA
      CE2=CE*CE
      SE2=SE*SE
C
C
      R(1,1)=R2*(EVAR*SE2*SA2+AVAR*CE2*CA2)+RVAR*CE2*SA2
      R(2,2)=R2*(EVAR*SE2*CA2+AVAR*CE2*SA2)+RVAR*CE2*CA2
      R(3,3)=R2*EVAR*CE2+RVAR*SE2
      R(1,2)=R2*EVAR*SE2*SA*CA+(RVAR-R2*AVAR)*(CE2*SA*CA)
      R(2,1)=R(1,2)
      R(1,3)=(RVAR-R2*EVAR)*SE*CE*SA
      R(3,1)=R(1,3)
      R(2,3)=(RVAR-R2*EVAR)*SE*CE*CA
      R(3,2)=R(2,3)
C
C
C
      FK1(1)=0.0
      FK1(2)=0.0
      FK1(3)=0.0
      FK1(4)=0.0
      FK1(5)=0.0
      FK1(6)=0.0
C
C
      IF (KAD .NE. 0) GO TO 29
         W(1,1)=W1
         W(2,2)=W2
         W(3,3)=W3
         GO TO 34
 29   CONTINUE
      IF (KAD .NE. 1) GO TO 31
         CALL ADPTV1(N,6,120,W1,W2,W3,TACREX,TACREY,TACREZ,
     +W)
         GO TO 34
 31   CONTINUE
         IF (TTG .LT. 20.) GO TO 33
            XRS=XRES(N-1)
            YRS=YRES(N-1)
            ZRS=ZRES(N-1)
            CALL ADPTV2(6,XRS,IXFLG,YRS,IYFLG,ZRS,IZFLG,PKK)
 33      CONTINUE
         W(1,1)=W1
         W(2,2)=W2
         W(3,3)=W3
 34   CONTINUE
      W11(N)=W(1,1)
      W22(N)=W(2,2)
      W33(N)=W(3,3)
C
C GENERATE Q-MATRIX.
      CALL QUADPS(DEL,W,NULL,6,6,Q)
C
C
C X(K/K-1)=PHI(K,K-1)*X(K-1/K-1)+F(K-1)
      DO 210 I=1,6
         XKK1(I)=FK1(I)
         DO 200 J=1,6
            XKK1(I)=XKK1(I)+PHI(I,J)*XKK(J)
```

```
 200        CONTINUE
 210     CONTINUE
C
C
C P(K/K-1)=PHI(K/K-1)*P(K-1/K-1)*TRANSPOSE(PHI)+Q(K-1)
         CALL QUADPS(PHI,PKK,Q,6,6,PKK1)
C
C
C G(K)=P(K/K-1)*H(K)*INV(H(K)*P(K/K-1)*TRANSPOSE(H)+R(K))
         CALL QUADPS(H,PKK1,R,6,6,EXTRA1)
         DO 212 I =1,3
            DO 211 J=1,3
               EXTRA4(I,J)=EXTRA1(I,J)
 211        CONTINUE
 212     CONTINUE
         DO 230 I=1,6
            DO 225 J=1,6
               EXTRA2(I,J)=0.0
               DO 220 K=1,6
                  EXTRA2(I,J)=EXTRA2(I,J)+PKK1(I,K)*H(J,K)
 220           CONTINUE
 225        CONTINUE
 230     CONTINUE
C
C
C
C INVERT EXTRA1 AND MULTIPLY BY EXTRA2.
         CALL LINV2F(EXTRA4,3,3,EXTRA3,IDGT,WKAREA,IER)
         CALL PRODCT(EXTRA2,EXTRA3,3,3,6,G)
         GX(N)=G(1,1)
         GY(N)=G(3,2)
         GZ(N)=G(5,3)
C X(K/K)=X(K/K-1)+G(K)*(Z(K)-H(K)*X(K/K-1))
         DO 250 I=1,6
            EXTRA1(I,1)=Z(I)
            DO 240 J =1,6
               EXTRA1(I,1)=EXTRA1(I,1)-H(I,J)*XKK1(J)
 240        CONTINUE
 250     CONTINUE
         RADRES(N)=SQRT(EXTRA1(1,1)**2+EXTRA1(2,1)**2+
     +EXTRA1(3,1)**2)
         XRES(N)=EXTRA1(1,1)
         YRES(N)=EXTRA1(2,1)
         ZRES(N)=EXTRA1(3,1)
         DO 270 I=1,6
            XKK(I)=XKK1(I)
            DO 260 J=1,3
               XKK(I)=XKK(I)+G(I,J)*EXTRA1(J,1)
 260        CONTINUE
 270     CONTINUE
         TACREX(1,N)=XKK(1)
         TACREX(2,N)=XKK(2)
         TACREY(1,N)=XKK(3)
         TACREY(2,N)=XKK(4)
         TACREZ(1,N)=XKK(5)
         TACREZ(2,N)=XKK(6)
C
C P(K/K)=P(K/K-1)-G(K)*H(K)*P(K/K-1)
         CALL PRODCT(G,H,6,6,6,EXTRA1)
         CALL PRODCT(EXTRA1,PKK1,6,6,6,EXTRA2)
         DO 290 I=1,6
            DO 280 J=1,6
               PKK(I,J)=PKK1(I,J)-EXTRA2(I,J)
 280        CONTINUE
 290     CONTINUE
C
 999     CONTINUE
```

```
      P(N,1)=PKK(1,1)
      P(N,2)=PKK(2,2)
      P(N,3)=PKK(3,3)
      P(N,4)=PKK(4,4)
      P(N,5)=PKK(5,5)
      P(N,6)=PKK(6,6)
C
      RETURN
      END
C
C
C
      SUBROUTINE KALMN2(KAD,DT,RSIGMA,ASIGMA,ESIGMA,RANGE,
     +AZMITH,ELEVTN,TTG)
C
C
C
      DOUBLE PRECISION DSEED
      COMMON/DOUBLE/DSEED
      COMMON/TMRRST/N,TACRRX(3,1000),TACRRY(3,1000),
     +TACRRZ(3,1000)
      COMMON/NOISE/TACRMX(1000),TACRMY(1000),TACRMZ(1000)
      COMMON/FILTER/TACREX(3,1000),TACREY(3,1000),
     +TACREZ(3,1000)
      COMMON/PROCES/X2SIGM,Y2SIGM,Z2SIGM,EWVRRX,EWVRRY,
     +GX(1000),GY(1000),GZ(1000),RADRES(1000),XRES(1000),
     +YRES(1000),ZRES(1000),W11(1000),W22(1000)
     +,W33(1000),P(1000,9),PINITL
C
      DIMENSION Q(9,9),PKK(9,9),R(9,9),W(9,9),G(9,9),
     +PHI(9,9),DEL(9,9),DI(9,9),NULL(9,9),H(9,9),XKK(9),
     +XKK1(9),FK1(9),EXTRA1(9,9),EXTRA2(9,9),PKK1(9,9)
     +,WKAREA(75),EXTRA3(3,3),EXTRA4(3,3),Z(9)
C
C
C
      IF (N .NE. 1 ) GO TO 25
      IDGT=2
      DT2=DT*DT/2.0
      DT3=DT2*DT/3.0
C
C
      DO 20 I=1,9
         Z(I)=0.0
         DO 10 J=1,9
            Q(I,J)=0.0
            PKK(I,J)=0.0
               IF (I .EQ. J) PKK(I,J)=PINITL
            R(I,J)=0.0
            W(I,J)=0.0
            G(I,J)=0.0
            PHI(I,J)=0.0
            DEL(I,J)=0.0
            DI(I,J)=0.0
               IF (I .EQ. J) DI(I,J)=1.0
            NULL(I,J)=0.0
            H(I,J)=0.0
 10      CONTINUE
 20   CONTINUE
      W1=X2SIGM*X2SIGM
      W2=Y2SIGM*Y2SIGM
      W3=Z2SIGM*Z2SIGM
      WX=0.0
      WY=WX
      WZ=WY
      IXFLG=0
      IYFLG=0
```

```
      IZFLG=0
      DEL(1,1)=DT3
      DEL(2,1)=DT2
      DEL(3,1)=DT
      DEL(4,2)=DT3
      DEL(5,2)=DT2
      DEL(6,2)=DT
      DEL(7,3)=DT3
      DEL(8,3)=DT2
      DEL(9,3)=DT
      H(1,1)=1.0
      H(2,4)=1.0
      H(3,7)=1.0
      PHI(1,1)=1.0
      PHI(1,2)=DT
      PHI(1,3)=DT2
      PHI(2,2)=1.0
      PHI(2,3)=DT
      PHI(3,3)=1.0
      PHI(4,4)=1.0
      PHI(4,5)=DT
      PHI(4,6)=DT2
      PHI(5,5)=1.0
      PHI(5,6)=DT
      PHI(6,6)=1.0
      PHI(7,7)=1.0
      PHI(7,8)=DT
      PHI(7,9)=DT2
      PHI(8,8)=1.0
      PHI(8,9)=DT
      PHI(9,9)=1.0
      RVAR=RSIGMA*RSIGMA
      AVAR=ASIGMA*ASIGMA
      EVAR=ESIGMA*ESIGMA
      TACREX(1,N)=TACRMX(1)
      XKK(1)=TACREX(1,N)
      TACREX(2,N)=TACRRX(2,N)+EWVRRX
      XKK(2)=TACREX(2,N)
      XKK(3)=TACRRZ(3,N)
      TACREY(1,N)=TACRMY(1)
      XKK(4)=TACREY(1,N)
      TACREY(2,N)=TACRRY(2,N)+EWVRRY
      XKK(5)=TACREY(2,N)
      XKK(6)=TACRRY(3,N)
      TACREZ(1,N)=TACRMZ(1)
      XKK(7)=TACREZ(1,N)
      TACREZ(2,N)=TACRRZ(2,N)
      XKK(8)=TACREZ(2,N)
      XKK(9)=TACRRZ(3,N)
C
      IF (N .EQ. 1) GO TO 999
C
C
C
 25   CONTINUE
C
C
C

      Z(1)=TACRMX(N)
      Z(2)=TACRMY(N)
      Z(3)=TACRMZ(N)
C
C

      R2=RANGE*RANGE
      CA=COS(AZMITH)
      SA=SIN(AZMITH)
      CE=COS(ELEVTN)
```

```
      SE=SIN(ELEVTN)
      CA2=CA*CA
      SA2=SA*SA
      CE2=CE*CE
      SE2=SE*SE
C
C
      R(1,1)=R2*(EVAR*SE2*SA2+AVAR*CE2*CA2)+RVAR*CE2*SA2
      R(2,2)=R2*(EVAR*SE2*CA2+AVAR*CE2*SA2)+RVAR*CE2*CA2
      R(3,3)=R2*EVAR*CE2+RVAR*SE2
      R(1,2)=R2*EVAR*SE2*SA*CA+(RVAR-R2*AVAR)*(CE2*SA*CA)
      R(2,1)=R(1,2)
      R(1,3)=(RVAR-R2*EVAR)*SE*CE*SA
      R(3,1)=R(1,3)
      R(2,3)=(RVAR-R2*EVAR)*SE*CE*CA
      R(3,2)=R(2,3)
C
C
C
      FK1(1)=0.0
      FK1(2)=0.0
      FK1(3)=0.0
      FK1(4)=0.0
      FK1(5)=0.0
      FK1(6)=0.0
      FK1(7)=0.0
      FK1(8)=0.0
      FK1(9)=0.0
C
C
      IF (KAD .NE. 0) GO TO 29
         W(1,1)=W1
         W(2,2)=W2
         W(3,3)=W3
         GO TO 34
 29   CONTINUE
      IF (KAD .NE. 1) GO TO 31
         CALL ADPTV1(N,9,120,W1,W2,W3,TACREX,TACREY,TACREZ,
     +W)
         GO TO 34
 31   CONTINUE
         IF (TTG .LT. 20.) GO TO 33
            XRS=XRES(N-1)
            YRS=YRES(N-1)
            ZRS=ZRES(N-1)
            CALL ADPTV2(9,XRS,IXFLG,YRS,IYFLG,ZRS,IZFLG,PKK)
 33      CONTINUE
         W(1,1)=W1
         W(2,2)=W2
         W(3,3)=W3
 34   CONTINUE
      W11(N)=W(1,1)
      W22(N)=W(2,2)
      W33(N)=W(3,3)
C
C GENERATE Q-MATRIX.
      CALL QUADPS(DEL,W,NULL,9,9,Q)
C
C
C X(K/K-1)=PHI(K,K-1)*X(K-1/K-1)+F(K-1)
      DO 210 I=1,9
         XKK1(I)=FK1(I)
         DO 200 J=1,9
            XKK1(I)=XKK1(I)+PHI(I,J)*XKK(J)
 200     CONTINUE
 210  CONTINUE
C
```

```
C
C P(K/K-1)=PHI(K/K-1)*P(K-1/K-1)*TRANSPOSE(PHI)+Q(K-1)
      CALL QUADPS(PHI,PKK,Q,9,9,PKK1)
C
C
C G(K)=P(K/K-1)*H(K)*INV(H(K)*P(K/K-1)*TRANSPOSE(H)+R(K))
      CALL QUADPS(H,PKK1,R,9,9,EXTRA1)
      DO 212 I =1,3
         DO 211 J=1,3
            EXTRA4(I,J)=EXTRA1(I,J)
 211     CONTINUE
 212  CONTINUE
      DO 230 I=1,9
         DO 225 J=1,9
            EXTRA2(I,J)=0.0
            DO 220 K=1,9
               EXTRA2(I,J)=EXTRA2(I,J)+PKK1(I,K)*H(J,K)
 220        CONTINUE
 225     CONTINUE
 230  CONTINUE
C
C
C
C INVERT EXTRA1 AND MULTIPLY BY EXTRA2.
      CALL LINV2F(EXTRA4,3,3,EXTRA3,IDGT,WKAREA,IER)
      CALL PRODCT(EXTRA2,EXTRA3,3,3,9,G)
      GX(N)=G(1,1)
      GY(N)=G(4,2)
      GZ(N)=G(7,3)
C X(K/K)=X(K/K-1)+G(K)*(Z(K)-H(K)*X(K/K-1))
      DO 250 I=1,9
         EXTRA1(I,1)=Z(I)
         DO 240 J =1,9
            EXTRA1(I,1)=EXTRA1(I,1)-H(I,J)*XKK1(J)
 240     CONTINUE
 250  CONTINUE
      RADRES(N)=SQRT(EXTRA1(1,1)**2+EXTRA1(2,1)**2+
     +EXTRA1(3,1)**2)
      XRES(N)=EXTRA1(1,1)
      YRES(N)=EXTRA1(2,1)
      ZRES(N)=EXTRA1(3,1)
      DO 270 I=1,9
         XKK(I)=XKK1(I)
         DO 260 J=1,3
            XKK(I)=XKK(I)+G(I,J)*EXTRA1(J,1)
 260     CONTINUE
 270  CONTINUE
      TACREX(1,N)=XKK(1)
      TACREX(2,N)=XKK(2)
      TACREX(3,N)=XKK(3)
      TACREY(1,N)=XKK(4)
      TACREY( 2,N)=XKK(5)
      TACREY(3,N)=XKK(6)
      TACREZ(1,N)=XKK(7)
      TACREZ(2,N)=XKK(8)
      TACREZ(3,N)=XKK(9)
C
C
C P(K/K)=P(K/K-1)-G(K)*H(K)*P(K/K-1)
      CALL PRODCT(G,H,9,9,9,EXTRA1)
      CALL PRODCT(EXTRA1,PKK1,9,9,9,EXTRA2)
      DO 290 I=1,9
         DO 280 J=1,9
            PKK(I,J)=PKK1(I,J)-EXTRA2(I,J)
 280     CONTINUE
 290  CONTINUE
 999  CONTINUE
```

```
      P(N,1)=PKK(1,1)
      P(N,2)=PKK(2,2)
      P(N,3)=PKK(3,3)
      P(N,4)=PKK(4,4)
      P(N,5)=PKK(5,5)
      P(N,6)=PKK(6,6)
      P(N,7)=PKK(7,7)
      P(N,8)=PKK(8,8)
      P(N,9)=PKK(9,9)
C
C15   FORMAT(' ','G(',I1,',',I1,')=',F20.10)
      RETURN
      END
C
C
C
      SUBROUTINE QUADPS(X,H,R,M,N,C)
         DIMENSION X(M,N),H(N,N),R(M,M),C(M,M)
            DO 400 I =1,M
               DO 300 J=1,M
                  C(I,J)=R(I,J)
                  DO 200 K=1,N
                     DO 100 L=1,N
                        C(I,J)=C(I,J)+X(I,K)*H(K,L)*X(J,L)
 100                 CONTINUE
 200              CONTINUE
 300           CONTINUE
 400        CONTINUE
            RETURN
         END
C
C
C
      SUBROUTINE ADPTV1(N,I3,JSTART,W1,W2,W3,TACREX,TACREY,
     +TACREZ,W)
         DIMENSION TACREX(3,1000),TACREY(3,1000),
     +TACREZ(3,1000),W(I3,I3)
         ISTATE=I3/3
         IF (N .LT. JSTART) GO TO 27
            WX=8.0*(TACREX(ISTATE,N-1)-TACREX(ISTATE,N-2))
            WY=8.0*(TACREY(ISTATE,N-1)-TACREY(ISTATE,N-2))
            WZ=8.0*(TACREZ(ISTATE,N-1)-TACREZ(ISTATE,N-2))
            GO TO 35
 27      CONTINUE
            WX=0.0
            WY=0.0
            WZ=0.0
 35      CONTINUE
         IF (ISTATE .EQ. 3) GO TO 37
            W(1,1)=W1+(WX*WX)/3.
            W(2,2)=W2+(WY*WY)/3.
            W(3,3)=W3+(WZ*WZ)/3.
         GO TO 41
 37      CONTINUE
            W(1,1)=W1+(WX*WX)/81.
            W(2,2)=W2+(WY*WY)/81.
            W(3,3)=W3+(WZ*WZ)/81.
 41      CONTINUE
         RETURN
      END
C
C
C
      SUBROUTINE ADPTV2(NORDER,XRES,IXFLG,YRES,IYFLG,ZRES,
     +IZFLG,PKK)
         DIMENSION PKK(NORDER,NORDER)
         RESET=1000.
```

```
         CALL BIAS(XRES,IXFLG)
         IF (IABS(IXFLG) .LT. 8) GO TO 10
            PKK(1,1)=RESET
            PKK(2,2)=RESET
            IF (NORDER .EQ. 9) PKK(3,3)=RESET
 10      CONTINUE
         CALL BIAS(YRES,IYFLG)
         IF (IABS(IYFLG) .LT. 8) GO TO 14
            IF (NORDER .EQ. 9) GO TO 12
               PKK(3,3)=RESET
               PKK(4,4)=RESET
               GO TO 14
 12         CONTINUE
            PKK(4,4)=RESET
            PKK(5,5)=RESET
            PKK(6,6)=RESET
 14      CONTINUE
         CALL BIAS(ZRES,IZFLG)
         IF (IABS(IZFLG) .LT. 8) GO TO 20
            IF (NORDER .EQ. 9) GO TO 16
               PKK(5,5)=RESET
               PKK(6,6)=RESET
               GO TO 20
 16         CONTINUE
            PKK(7,7)=RESET
            PKK(8,8)=RESET
            PKK(9,9)=RESET
 20      CONTINUE
         RETURN
      END
C
C
      SUBROUTINE BIAS(RESIDU,IFLAG)
         IF (RESIDU) 10,20,30
 10         IF (IFLAG .LE. 0) GO TO 12
               IFLAG=0
               GO TO 40
 12         CONTINUE
               IFLAG=IFLAG-1
 14         GO TO 40
 20         IFLAG=0
            GO TO 40
 30         IF (IFLAG .GE. 0) GO TO 32
               IFLAG=0
               GO TO 40
 32         CONTINUE
            IFLAG=IFLAG+1
 40      CONTINUE
         RETURN
      END
```

LIST OF REFERENCES

1. Lentz, R. E., Improvement of AN/TPQ-27 Filter and Control Techniques, Naval Postgraduate School, 1974.

2. Singer, R. A., Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets, IEEE Trans. AES, Vol. 6, No. 4, July 1970.

3. Benedict, T. R. and Bordner, G. W., Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations, IRE Transactions on Automatic Control, February 1962.

4. Quigley, Anthony L. C., Development of a Simple Adaptive Target Tracking Filter and Predictor for Fire Control Applications, Naval Surface Weapons Center, Dalgren, Virginia, December 1978.

5. Clark, B. L., Development of an Adaptive Kalman Target Tracking Filter and Predictor for Fire Control Applications, Naval Surface Weapons Center, Dalgren, Virginia, March 1977.

6. Aldrich, G. T. and Krabill, W. B., An Application of Kalman Techniques to Aircraft and Missile Radar Tracking, Paper 72-838, AIAA Guidance and Control Conference, Stanford, California, August 1972.

7. Mayoral, L. M., An Adaptive Kalman Identifier and Its Application to Linear and Non-Linear ARMA Modeling, Naval Postgraduate School, 1981.

200023

Thesis
J337 Jauregui
c.1 Aircraft state
estimation for a
ground directed bombing
system.

200023

Thesis
J337 Jauregui
c.1 Aircraft state
estimation for a
ground directed bombing
system.

thesJ337
Aircraft state estimation for a ground d
3 2768 001 02497 9
DUDLEY KNOX LIBRARY